MACHINE LEARNING ALGORITHM DESIGN FOR HARDWARE PERFORMANCE OPTIMIZATION

A Dissertation
Presented to
The Academic Faculty

By

Shaojie (Kyle) Xu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2020

MACHINE LEARNING ALGORITHM DESIGN FOR HARDWARE PERFORMANCE OPTIMIZATION

Approved by:

Dr. Justin Romberg, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Arijit Raychowdhury
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Hua Wang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Eva Dyer
Wallace H. Coulter Department of Biomedical Engineering
Georgia Institute of Technology and Emory University

Date Approved: March 20, 2020

ACKNOWLEDGEMENTS

I have received tremendous help from many professors, colleagues, friends, and family members during my journey as a Ph.D. student. Over the past six years, I have had significant growth both professionally and personally. I owe much of this growth to the many people who have provided me support, guidance, and love.

I would like to pay the greatest tribute to my advisor, Dr. Justin Romberg. His thoughtful guidance, firm support, and continuous encouragement have been invaluable to me during my graduate school years. The completion of my dissertation would not have been possible without his professional help and inspiration. His attitude, hard work, and dedication have fundamentally shaped how I conduct research.

I would also like to express my sincere thanks to Drs. Arijit Raychowdhury and Hua Wang for collaborating with me on research projects. Their extensive knowledge and insightful guidance are indispensable for the success of the projects. I must thank Drs. Mark Davenport and Eva Dyer for reviewing my thesis and providing profound suggestions.

Special thanks to my fellow student collaborators Fei Wang, Sihan Zeng, and Anvesha Amaravati. The research projects are only made possible with their hard work and experties. I have also been helped by outstanding people in our joint research group Children of the Norm. The discussions among group members have widened my horizons.

I owe special gratitude to my mentors outside the Georgia Tech community: Aidin Bassam, Sherif Shakib, and Paul Draxler from Qualcomm, Ioseph Martinez and Janna Garofolo from Freescale, Young Ho Cha from ExxonMobil, and Svitlana Vyetrenko from J.P. Morgan. They have provided me valuable insights and perspectives on industrial research.

Finally, I am forever indebted to my parents, my grandparents, and other family members. Their unconditional love and support is the source of my strength and dedication.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The objective of this thesis is to combine techniques in machine learning, signal processing, and system control for hardware performance optimization. By leveraging collected data to construct a better model for both the hardware and the operating environment, machine learning enables the hardware to operate more power-efficiently, to obtain improved results, and to maintain robust performance against environmental changes. The proposed work targets three aims: (i) design data-driven signal processing algorithms which require fewer measurements taken from the sensor front-end; (ii) develop algorithm-hardware co-design techniques for hardware that performs specific machine learning tasks; (iii) design adaptive hardware control algorithms.

For the first aim, we propose an algorithm for image reconstruction from the compressed measurements with image priors captured by a generative model. We search in the generative model's latent variable space to make the method stable when the number of compressed measurements is extremely limited. We show that, by exploiting certain structures of the latent variables, the proposed method produces improved reconstruction accuracy and preserves realistic and non-smooth features in the image. Our algorithm achieves high computation speed by projecting between the original signal space and the latent variable space in an alternating fashion.

For the second aim, we propose a novel appearance-based gesture recognition algorithm using compressed domain signal processing techniques. Gesture features are extracted directly from the compressed measurements, which are the block averages and the coded linear combinations of the image sensors pixel values. We also improve both the computational efficiency and the memory requirement of the previous gesture classifiers. Both simulation testing and hardware implementation strongly support the proposed algorithm.

For the third aim, we propose several Doherty power amplifier (PA) control algorithms based on reinforcement learning and the bandit framework. Our algorithms achieve robust

adaptive operation over environmental changes. Multiple bandit frameworks are incorporated in the control algorithms including multi-armed bandit, continuum-armed bandit, contextual-bandit, and actor-critic with experience replay. The latter three algorithms leverage on the prior information about the Doherty PA's characteristics to improve learning efficiency.

CHAPTER 1
INTRODUCTION

Machine learning has enabled us to extract and exploit information from collected data. In this thesis, we are particularly interested in how we can apply this powerful tool to enhance the performance of various hardware. We aim to explore the application of machine learning from three different angles. In the first angle, machine learning is used to improve the processing step after the signals being acquired by the sensor front-ends. One important criterion when designing sensor front-ends is their energy efficiency. Modern systems such as sensor networks and internet of things (IoT) deploys a large number of sensors that continuously collecting data. Reducing the number of measurements while maintaining satisfactory result after the signal processing step can lead to significant energy saving. In other applications such as medical imaging and seismic imaging, the data acquisition step is laborious. Designing signal processing algorithms that require fewer measurements can largely reduce the data acquisition cost.

Recent developments in compressive sensing [1, 2, 3] have improved the performance and energy efficiency of the overall process of data acquisition. The linear compression in this framework may naturally come from the sensing scheme itself, as in the case of computed tomography (CT) and magnetic resonance imaging (MRI), or come from an imposed random projection step. The processing step retrieves the original signal from the compressed measurements. Therefore, in the first aim of this thesis, we combine machine learning and compressive sensing to design data-driven compressive sensing recovery algorithms. We will show that by capturing the structure of the signal directly from the dataset, data-driven compressive sensing recovery algorithms reduce the number of required measurement and improve the recovery quality. Furthermore, the recovery process can be accelerated under the help of machine learning.

In our first aim, the signal processing step and the data acquisition step are separated. The acquisition scheme is implemented in the circuit level while the signal processing algorithm is implemented in the software level. In the second aim of this thesis, the hardware itself is designed to perform certain machine learning tasks. The algorithm, therefore, is integrated as part of the hardware and is realized in the mixed-signal domain. Specifically, we propose a system that uses motion gestures as stimuli to "wake up" [4, 5, 6]. To achieve power efficiency, the system turns into the idle mode when not being used and wakes up when users are detected. This system can be regarded as a gesture equivalent to Amazon's popular Echo devices, which wake up after "hearing" the key word "Alexa".

The detection of an user's present requires the sensor front-ends to be perpetually on, thus making low power consumption an important design criteria. Traditional appearance-based gesture recognition algorithms consume a significant amount of energy in analog to digital conversion of each pixel of the image sensor. Our proposed gesture recognition algorithm again combines compressive sensing and machine learning. Our sensor front-end takes coded combinations of the pixel vales and the algorithm characterizes the gesture motion directly from a few compressed measurements. Combining compressive sensing and machine learning into a hardware-algorithm co-design sheds new lights on developing sensor networks and internet of things for particular learnable tasks [7, 8].

The integration of the machine learning algorithm and hardware is more prominent in our third aim, where we design an adaptive hardware control algorithm using the bandit framework from reinforcement learning (RL). The hardware we focus here is a power amplifier (PA) used in 5G telecommunication. The control unit observes the PA's inputs and outputs, and uses them to characterize the PA's behavior under various control settings. As the PA's operating environment changes over time, the control unit needs to continuously learn the new optimal control policy. The close relationship between machine learning algorithms and the hardware design will be well demonstrated in this project. On one hand, the control algorithm adjusts the control policy timely and allows the PA to maintain robust

performance against environmental changes. On the other hand, the control algorithm must rely on the prior knowledge about how the hardware behaves under different settings to achieve the best learning efficiency.

The rest of this chapter provides technical background for the projects presented in this thesis. The discussion in this chapter is limited to a high level overview. Detailed discussions and the start-of-the-art methods related to each project are included in the corresponding chapters. A highlight of our contributions is also provided in this chapter. Chapter 2, 3,4 describe the projects corresponding to our three aims respectively. As a wide range of projects is presented in this thesis, it is almost inevitable to overuse certain math symbols cross different chapters. However, the use of math symbols within in each chapter and within each project should be clear and consistent. Chapter 5 concludes the whole thesis with discussions about some related open problems at the time of writing this thesis.

1.1 Background

1.1.1 Linear inverse problem and compressive sensing

Linear inverse problem assumes the signal of interest $\boldsymbol{x} \in \mathbb{R}^N$ and its measurements $\boldsymbol{y} \in \mathbb{R}^M$ satisfy the relationship:

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x} + w \ , \tag{1.1}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ is the measurement matrix and $w \in \mathbb{R}^M$ is the noise. w is assumed to be zero-mean and independent of $\boldsymbol{x}$. Many modern imaging systems and computational imaging problems can be described by this formulation. For example, in CT, the measurement matrix $\boldsymbol{\Phi}$ implements the Radon transform which integrates the image along lines of different directions. In super-resolution problems, $\boldsymbol{\Phi}$ is the down-sampling matrix. In image deblurring problems, $\boldsymbol{\Phi}$ performs convolution on the original image with a blurring kernel.

Our specific interest is to the filed of compressive sensing, where we seek to reconstruct a high-dimensional signal after observing a small number of its random linear encodings.

The compression is achieved as we require the number of measurements to be much smaller than the number of elements contained in the original signal: $M \ll N$. The measurment matrix contains randomeness. Common methods to construct $\mathbf{\Phi}$ include choosing each entry from an independent and identically distributed (i.i.d.) Gaussian [9], a fair Bernoulli, or an independent Sub-Gaussian distribution [10].

Both the compressive property and the randomness contained in the measurement matrix are import for compressive sensing to be implemented in the hardware design. The compressive property allows the sensing hardware to obtain much fewer measurements and, therefore, save energy from data acquisition. One may argue that the linear operation used to achieve this compression could be computationally expensive and cancel out all the energy savings from the sensor front-end. However, in many of the imaging system such as CT and MRI, the linear compression is already part of the natural scheme. And in the case of requiring manual linear compression, it has been show that much of the energy consumption in the imaging system comes from the analog to digital conversion (A/D) [4, 5]. The linear compression operation can be implemented efficiently in the analog or mixed-signal domain. Significant energy saving is achieved from fewer A/D operations at the output end of the sensor.

The randomness in $\mathbf{\Phi}$ allows computational and memory efficient implementations of the compression scheme. For instance, we can construct a $\mathbf{\Phi}$ by sampling each entry from an i.i.d Bernoulli distribution with value -1 and 1. $\mathbf{\Phi}$ can be stored as a binary matrix, and only simple additions and subtractions are needed to calculate the compressed measurements. We will later see that the randomness in $\mathbf{\Phi}$ also plays an important role in preserving useful information from the original signal. In addition, it provides us a delicate way to derive theoretical recovery guarantees of compressive sensing.

In the case of useful compression, for two distinct signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the original space, their corresponding representations $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ in the compressed domain also need to be separate. As $\mathbf{\Phi}$ is an under-determined matrix with a null space, information can not be

preserved for all vectors in $\mathbb{R}^N$. Fortunately, real-life signals often have prominent structures and only lie in some subspace of $\mathbb{R}^N$. Traditional compressive sensing focuses on sparse signals. When a signal $\boldsymbol{x}$ has (or approximately has) a k-sparse representation in some basis, the compression matrix $\boldsymbol{\Phi}$ can be constructed to preserve the distances between two signals in the compressed domain, a property formally known as the restricted isometry property (RIP) [2]. Similar results were also established for signals on a smooth manifold [11].

Given the compressed measurements $\boldsymbol{y}$, there are two types of tasks can be performed. The first type is to recover the original signal $\boldsymbol{x}$. In the case of $M < N$, the null space of $\boldsymbol{\Phi}$ makes the recovery problem ill-posed: the true $\boldsymbol{x}$ plus any vector in the null space of $\boldsymbol{\Phi}$ results in the same compressed measurements. Therefore, in order to pick the signal of interest from all others that are corrupted by null space vectors, we need to rely on some prior knowledge about the true signal. As mentioned before, sparsity and smooth manifold are among some of the common prior knowledge assumed for recovery. For some applications, recovering the whole signal is not necessary. For example, if we want to classify dog versus cat images, it may be unnecessary to recover all the details in the pictures before running the classification algorithm. It is likely that cats and dogs have different representations directly in the compressed domain. In the gesture recognition application to be discussed in Chapter 3, we try to estimate the location of the hand from each compressed video frame. Recovering the details about a hand is not important as long as its location in each video frame can be accurately estimated. All of these are examples of the second type of task which to extract features or estimate parameters directly from the compressed measurements.

1.1.2 Compressive sensing recovery

In order to retrieve the original signal from the compressed measurements, some type of prior knowledge about the original signal needs to be assumed in order to remedy the ill-posed inverse problem. When the original signals are well-studied, the prior knowledge comes from years of experience. For example, the sparsity model is often used to recover

compressively sensed natural images. Now with the power of machine learning, we can extract prior knowledge directly from collected datasets.

When we have explicit formulation of the subspace where the original signal is believed to come from, the recovery process can be formulated as an optimization problem of the following form:

$$\min_{\boldsymbol{x}\in\mathcal{S}} \; F_{\boldsymbol{y}}(\boldsymbol{x}) \; . \tag{1.2}$$

The function $F_{\boldsymbol{y}}(\boldsymbol{x})$ controls how well a candidate signal $\boldsymbol{x}$ matches the measurement $\boldsymbol{y}$ in the compressed domain. With the compressive sensing formulation in (1.1), the measurement $\boldsymbol{y}$ of an original signal $\boldsymbol{x}$ is a random variable following the w's distribution with a shifted mean. This statistical view leads to design $F_{\boldsymbol{y}}(\boldsymbol{x})$ to measure $P(\boldsymbol{y}|\boldsymbol{x})$, the probability of observing $\boldsymbol{y}$ under some original signal $\boldsymbol{x}$. This method is known as the maximum likelihood estimation (MLE). When w follows a Gaussian distribution, the MLE's objective function corresponds to measuring the Euclidean distance in the compressed domain: $F_{\boldsymbol{y}}(\boldsymbol{x}) = ||\boldsymbol{y} - \boldsymbol{\Phi x}||_2^2$.

The set $\mathcal{S}$ explicitly describe the subspace where the original signal lives in. For example, $\mathcal{S}$ can be a set of all K-sparse signals, a union of linear subspace, or a manifold. To solve (1.2) is to search candidate signals only in $\mathcal{S}$ and to find the one that minimizes the objective function. As our search is now confined in an embedded in $\mathbb{R}^N$, the originally ill-posed linear inverse problem is much likely to return satisfactory solutions as long as the nullspace of $\boldsymbol{\Phi}$ does not collapse this embedding. In general, (1.2) is a constrained optimization problem. However, if $\mathcal{S}$ can be written in the form of $\mathcal{S} = \{G(\boldsymbol{z}) \mid \boldsymbol{z} \in \mathbb{R}^L\}$, then (1.2) can be reduced to an unconstrained optimization problem. For example, if $\boldsymbol{x}$ is assumed to come from an L-dimensional linear subspace $\mathcal{S} = \{\boldsymbol{Bz} \mid \boldsymbol{z} \in \mathbb{R}^L\}$ or a non-linear generative model whose input space is $\mathbb{R}^L$, then we may be able to directly solve $\min_{\boldsymbol{z}} \; F_{\boldsymbol{y}}(G(\boldsymbol{z}))$.

In some cases, we do not have an explicit formulation of the original signals' subspace. Instead, we have a function $J(\boldsymbol{x})$ that measures how likely a candidate $\boldsymbol{x}$ comes from the

subspace of our interest. We can formulate the compressive sensing recovery problem as:

$$\min_{\boldsymbol{x}} \; F_{\boldsymbol{y}}(\boldsymbol{x}) + J(\boldsymbol{x}) \; . \tag{1.3}$$

$J(\boldsymbol{x})$ is chosen to promote certain properties the original signal should satisfy. For example, if $\boldsymbol{x}$ is sparse, $J(\boldsymbol{x})$ can be $||\boldsymbol{x}||_0$ or the convex relaxation form $||\boldsymbol{x}||_1$. As (1.2) can be associate with the MLE, (1.3) can be associated with maximum a posteriori estimation (MAP). We can use the Bayes' theorem to write the probability of $\boldsymbol{x}$ after observing $\boldsymbol{y}$:

$$P(\boldsymbol{x}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{x})P(\boldsymbol{x})}{P(\boldsymbol{y})} \; . \tag{1.4}$$

To search for $\boldsymbol{x}$ that maximizes $P(\boldsymbol{x}|\boldsymbol{y})$ is to search $\boldsymbol{x}$ that maximizes the product of $P(\boldsymbol{y}|\boldsymbol{x})$ and $P(\boldsymbol{x})$. Converting (1.4) to log probabilities, we have the Bayesian formulation of compressive sensing recovery:

$$\min_{\boldsymbol{x}} \; -\log(P(\boldsymbol{y}|\boldsymbol{x})) - \log(P(\boldsymbol{x})) \; . \tag{1.5}$$

One can compare this formulation with (1.3). We have shown that $F_{\boldsymbol{y}}(\boldsymbol{x})$ can be designed to reflect $-\log(P(\boldsymbol{y}|\boldsymbol{x}))$. Specifically, when the additive observation noise is assume to follow a Gaussian distribution, we use $F_{\boldsymbol{y}}(\boldsymbol{x}) = ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}||_2^2$. The signal property term $J(\boldsymbol{x})$ can be viewed to measure $-\log(P(\boldsymbol{x}))$. For example, $J(\boldsymbol{x}) = ||\boldsymbol{x}_1||$ is equivalent to assuming that $\boldsymbol{x}$ follows a Laplace distribution.

A large number of compressive sensing recovery algorithms under various prior assumptions has been proposed in the past decade. Detailed discussion about these algorithms are provided in Chapter 2. Our proposed framework in Chapter 2 follows the general formulation (1.2). The signal is assumed to come from a generative neural network $G(\boldsymbol{z})$. More background information about the generative model is provided in the next section. The recovery is carried out by running an iterative optimization method named alternating

direction method of multiplier (ADMM). Within each iteration of ADMM, we use another neural network to perform the the operation. Our new framework leads to faster recovery process with improved recovery quality, which allows wider implementation of compressive sensing in various hardware.

1.1.3 Generative neural networks

We have seen the importance of the prior knowledge in compressive sensing recovery. With the power of machine learning, we can extract prior knowledge directly from collected datasets. The prior knowledge can take the form of a distribution. In this way, we learn to approximate the distribution $P(\boldsymbol{x})$ and use it with (1.3). For example, we can learn a Gaussian mixture model (GMM) to describe the distribution of $\boldsymbol{x}$ and use its negative log probability as $J(\boldsymbol{x})$. The learned prior knowledge can also describe the original signal's space explicitly. In this way, a mapping $G(\cdot) : \mathbb{R}^L \to \mathbb{R}^N$ is learned to map a low dimensional subspace to the high dimensional original signal space. $\boldsymbol{x}$ is assumed to satisfy $\boldsymbol{x} \in \{G(\boldsymbol{z}) \mid \boldsymbol{z} \in \mathbb{R}^L\}$. This form of prior knowledge can be naturally incorporated into (1.2).

As the original signals often contain complicated structure, the mapping $G(\cdot)$ is required to have large modeling capacity. The recent developments in neural networks allow us to design powerful models that satisfy this requirement. We can start with a Gaussian random variable $\boldsymbol{z} \in \mathbb{R}^L$ with zero mean and identity covariance matrix. We aim to learn a mapping $G(\cdot;\theta)$ in the form of a neural network parameterized by θ, whose outputs $G(\boldsymbol{z};\theta)$ closely resemble the ones in the dataset of $\boldsymbol{x}$. What comes next is to design objective functions that both encourage the matching and can provide gradient updates for the neural network.

Variational autoencoder (VAE) [12] aims to maximize the probability of observing the signals in the dataset:

$$\max_{\theta} \sum_{i} \log(P(\boldsymbol{x}_i)) = \max_{\theta} \sum_{i} \log(\int P(\boldsymbol{x}_i|G(\boldsymbol{z};\theta))P(\boldsymbol{z})dz) \; . \tag{1.6}$$

However, evaluating the above objective function requires to integrate over all possible $\boldsymbol{z}$, making the problem intractable. One can realize that for a given $\boldsymbol{x}_i$ and a randomly sampled $\boldsymbol{z}$, $P(\boldsymbol{x}_i|G(\boldsymbol{z};\theta))$ is likely to be extremely small. Hence, a large portion of the integral in (1.6) is over small values. The solution offered by VAE is to assume a Gaussian distribution on $P(\boldsymbol{z}|\boldsymbol{x})$ and use another neural network to approximate the mean and the covariance matrix of this Gaussian distribution. We can sample from this proposed Gaussian distribution and approximate the integral with a finite number of samples. The discrepancies caused by this finite sample approximation and the $P(\boldsymbol{z}|\boldsymbol{x})$ approximation are properly handled in VAE.

Another type of generative neural networks is generative adversarial net (GAN) [13, 14]. GAN aims to match between the distribution of the generative network's (generator's) output and the distribution of the true signal's. A neural network named discriminator is constructed to judge how well the two distribution match each other. The training process is set up as the game between the generator and the discriminator. The discriminator tries to distinguish between the samples drawn from the true dataset and the samples generated by the generator. The generator, on the other hand, tries to fool the discriminator by generating samples as similar to the true data as possible. By constructing a discriminator $D(\boldsymbol{x};\theta_D)$ to represent the probability that $\boldsymbol{x}$ comes from the true dataset, the objective of GAN can be written as:

$$\min_{\theta_G} \max_{\theta_D} \mathrm{E}_{\boldsymbol{x}}[\log(D(\boldsymbol{x};\theta_D))] + \mathrm{E}_{\boldsymbol{z}}[\log(1 - D(G(\boldsymbol{z};\theta_G);\theta_D))] \,. \tag{1.7}$$

The generative neural networks we discussed so far are lack of both explainability and controllability. The latent variables, being random variables themselves, do not have clear semantic meanings with respect to the generated signals. Although [14] shows that the combinations of the latent variables of different type of images can be used to generate synthesized images, the correspondence between the type of image and the latent variable

is still unclear. For example, for a GAN trained on facial images, we can construct a new latent variables by adding the latent variables corresponding to men with glasses to the latent variables corresponding to women without classes, then minus the latent variables corresponding to men without glasses. These constructed latent variables can produce images of women with glasses. However, we have no explicit description on what type of latent variable corresponds to facial images with glasses. In fact, one type of image is often associate with some interaction among each element in the latent variable, an interaction that is too complicated to be untangled. The inability to associate between latent variables to their generated signal also prevents us from sampling the signal space in a controllable manner. Using the previous example, it is not clear how we navigate through the latent variable space to sample only facial images with glasses.

A new type of nerual network named InfoGAN [15] was proposed to solve the issues of explainability and controllability. InfoGAN first divide the latent variables into two parts: the semantic codeword $\boldsymbol{c}$ and the "random-noise-like" variable $\boldsymbol{v}$. A third neural network is constructed to infer back the codewords used by the generator to create corresponding images. The generator, therefore, not only tries to fool the discriminator, but also tries to produce images in a predicable way using the codewords. Figure 1.1 shows the MNIST digits generated by the InfoGAN using different codeword $\boldsymbol{c}$ while fixing the randomness variable $\boldsymbol{v}$. The codeword contains one categorical variable in the form of one-hot vector and two continuous variables. Each row in the figure corresponds to one category, which successfully learned to control the digit class. Images from left to right correspond to changing the two continuous variables from -1s to 1s. It is clear that these two continuous variables learned to control the tilted angle and thickness of the stroke. As a comparison, we also generated digits using fixed codeword $\boldsymbol{c}$ and randomly sampled $\boldsymbol{v}$. The results are shown in Figure 1.2. Each row corresponds to one categorical codeword while $\boldsymbol{v}$ is randomly sampled from left to right. $\boldsymbol{v}$ adds details and variations to the generated images, but its contribution is limited. Notice that the training of the InfoGANs is unsupervised. The

correspondence between the codeword and the semantic meaning in the generated signals is discovered by model itself. In the cases where labels on the signals are available, other models such as [16] have been proposed to enforce the association between the labels and the latent variables.

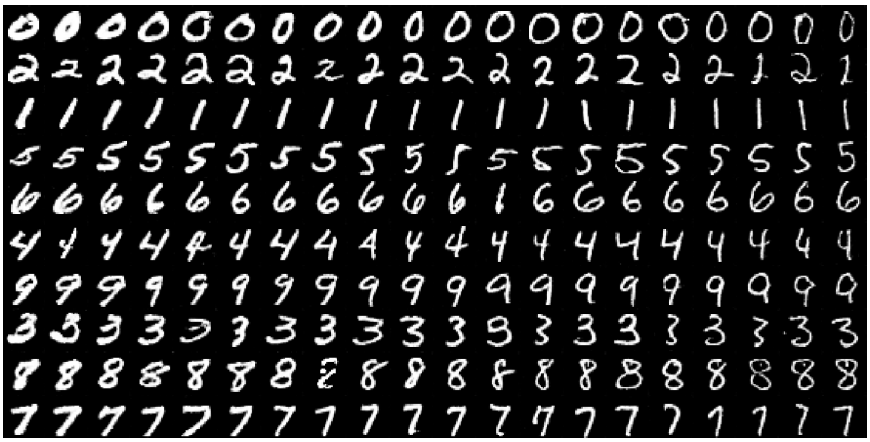


Figure 1.1: MNIST digits generated by an InfoGAN. Each row corresponds to one categorical codeword while two continuous variables change from -1s to 1s from left to right.

Figure 1.2: MNIST digits generated by an InfoGAN. Each row corresponds to one categorical codeword while randomness variable $\boldsymbol{v}$ is randomly sampled from left to right.

Separating the latent variable into codeword and randomness variable imposes structure in the generative model's latent variable space. The codeword which controls the basic information often lies in a much smaller subspace. As we will see in Chapter 2, this structured latent variable space allows us to reduce the number of compressed measurements

if only the basic form of the signal needs to be recovered. As the energy consumption of the sensor front-end is usually directly associated with the number of measurements, generative models with structured latent variable space improves the energy efficiency of the sensor front-end.

1.1.4 Compressed domain parameter estimation

Some machine learning tasks can be performed directly on the compressed measurements $\boldsymbol{y}$ without explicitly recovering the original signal $\boldsymbol{x}$. One of the areas that has shown early success is feature extraction and parameter estimation. Assume that the original signal $\boldsymbol{x}$ comes from a lower dimensional embedding in $\mathbb{R}^N$ which can be described by a parameterized function $f(\boldsymbol{\theta}),\ \boldsymbol{\theta} \in \mathbb{R}^K$:

$$\boldsymbol{x} = f(\boldsymbol{\theta}) + \omega \ , \tag{1.8}$$

where ω is an independent additive noise. If ω is assumed to follow a Gaussian distribution, the MLE of the parameter $\boldsymbol{\theta}$ given observation $\boldsymbol{x}$ is formulated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||\boldsymbol{x} - f(\boldsymbol{\theta})||_2^2 \ . \tag{1.9}$$

When we only observed the compressed measurements $\boldsymbol{y}$ which satisfies

$$\boldsymbol{y} = \boldsymbol{\Phi}(f(\boldsymbol{\theta}) + \omega) + w \ , \tag{1.10}$$

and assume w is also an independent additive Gaussian noise, then the MLE of $\boldsymbol{\theta}$ in the compressed domain is:

$$\hat{\boldsymbol{\theta}}^* = \arg\min_{\boldsymbol{\theta}} ||\boldsymbol{y} - \boldsymbol{\Phi} f(\boldsymbol{\theta})||_2^2 \ . \tag{1.11}$$

When the embedding $f(\cdot)$ has a nice compressible structure and the number of compressed measurements is sufficient, the solutions in (1.9) and (1.11) have a high probability

of being very close to each other. This result is a direct extension of the embedding's RIP which preserves the pairwise distance among points in the embedding. The RIP proof of smooth manifolds is presented in [17, 11]. The RIP proof of Lipschitz continuous mappings $f(\cdot)$ is provided in [18] and in Chapter 2 of this thesis.

In some applications, we do not have the explicit form of $f(\cdot)$, but rather discrete samples of this function. Solving (1.9) and (1.11) is equivalent to the nearest neighbor methods in the machine learning literature. In some applications, $f(\theta)$ forms a set of signals that are shifted versions of one another. In these cases, solving (1.9) is known as matched filtering and can be implemented efficiently with convolution. The compressed domain equivalence (1.11) is named "smashed filtering" in [17]. However, at the time of writing this thesis, its efficient implementation is still an open question. Performing machine learning tasks using data that are randomly projected into the compressed domain is the foundation of many randomized algorithms, whose wide applications are discussed in details in [19, 20].

In the motion gesture recognition project to be discussed in Chapter 3, each video frame is compressively measured. The first task to characterize the motion is to estimate the hand location within each video frame. The performance guarantee of estimating parameters directly in the compressed domain saves us the computation and energy otherwise needed for signal recovering. We observe that the compression rate to achieve reliable parameter estimation can often be much larger than to achieve good recovery, which encourages the use of compressive sensing in task-specific hardware.

1.1.5 Bandit problems

Multi-armed bandits: We now introduce a new area of machine learning that concerns decision making. Consider a bandit machine with N_a arms. These N_a arms form a set $\mathcal{A}$, and at each time t, the player chooses to pull one of the arms $a_t \in \mathcal{A}$. The chosen arm returns a stochastic reward r_t is drawn from an unknown distribution $R(a_t)$. The reward drawing process only depends on the currently selected arm and is independent of time. The

reward distribution is also assumed to be stationary and does not change with time. The player seeks to maximize his total expected reward summarized over a finite number of time steps. When one arm is pulled, the player gains some knowledge about the rewards distribution associated with that arm. In order to maximize the total expected reward, the player should keep track the expected reward of each arm based on all the reward samples he has previously collected from that arm:

$$\hat{R}_t(a) = \frac{\sum_{\tau=1}^{t} r_\tau \mathbf{1}_a(a_\tau)}{\sum_{\tau=1}^{t} \mathbf{1}_a(a_\tau)} \,, \tag{1.12}$$

where $\mathbf{1}_a(a_t)$ is the indicator function that equals to 1 when $a_t = a$ and 0 otherwise. This formula requires storing all the received rewards and arm choices in the memory. The estimation of the expected reward can be updated incrementally with memory efficient implementation:

$$\begin{aligned} \hat{R}_t(a_t) &= \frac{(N_t(a_t) - 1)\hat{R}_{t-1}(a_t) + r_t}{N_t(a_t)} \\ &= \hat{R}_{t-1}(a_t) + \frac{1}{N_t(a_t)} \left(r_t - \hat{R}_{t-1}(a_t) \right) \,, \end{aligned} \tag{1.13}$$

where $N_t(a_t) = \sum_{\tau=1}^{t} \mathbf{1}_{a_t}(a_\tau)$ is the total number of times that arm a_t has been pulled till t. The equation above provides us a new way to interpret the update of the expected reward estimation. When the player receives a new reward r_t from pulling the arm a_t, the previous estimation of the expected reward is updated by a portion of the difference between the newly received reward and the previous estimation. How much of this difference is added to the previous estimation is inversely proportional to the number of samples on this arm. As the number of samples increases, the contribution of each sample to the estimated expected reward decreases. The estimation converges to the true expected reward asymptotically.

If the player has unlimited number of steps, then he can choose to sample each arm many times to estimate the expected reward of each arm. Once the player is highly confident about the estimation, he can focus on playing the arm that has the highest estimated expected

reward. However, when the number of steps are limited, the player constantly faces two choices: to play the arm which has the highest estimated expected reward so far or to sampling another arm. The former choice is known as exploitation. Here the player chooses the best action according to his current knowledge about the bandit machine. The latter choice is known as exploration. Here the player spends the limit budget in gathering more information about other actions. The balance between exploration and exploitation is intuitive. At the beginning of the game, the player should explore many actions to avoid narrow and myopic believes on a small subset of actions. As the game advances, the growing number of samples increases the player's confidence about the estimation of each arm's expected reward. He, therefore, should shift his focus to exploitation and only pulls the arms from which he is confident about receiving high rewards.

The foundation of solving bandit problems is balancing between exploitation and exploration. One of the simplest methods is the ϵ-greedy algorithm [21]. At each step, with probability $1-\epsilon$, the player pulls the best arm based on his current estimation of the expected rewards of each arm. With probability ϵ, the player randomly choose one of the other arms to pull. During this exploration stage, each candidate arm (every arm except the best arm according to the player's current estimation) has an equal probability of being chosen. As ϵ is fixed, a large ϵ allows the player to find the best arm quickly. However, a large amount of exploration after gaining good knowledge of all the arms results in sub-optimal cumulative rewards. On the contrary, a small ϵ encourages the player to stick with the true optimal arm in the later part of a trial. However, it can take many steps before the player finds this true optimal arm and results in sub-optimal performance at the beginning of a trial.

As our intuition suggests, the probability of choosing one arm depends on two criteria: how this arm's estimated expected reward compares to other arms, and how confident we are about this estimation. Both the estimation and our confidence change when more reward samples are collected along the trial, which suggests an adaptive exploration rate. One algorithm to achieve this adaptive balancing scheme is the upper-confidence-bound (UCB)

algorithm [22]. In this algorithm, each arm has an associated UCB which is determined by both the estimated expected reward and by the number of times this arm has been pulled. The relative UCB increases when the estimated expected reward increases. The relative UCB decreases when its associated arm has been pulled more often compared to other arms. One specific UCB formulation is shown in Equation (1.14):

$$UCB(a) = \hat{R}_t(a) + c\sqrt{\frac{\ln(t+1)}{N_t(a)+1}}\,. \tag{1.14}$$

The parameter c controls how much the uncertainty term contributes to the UCB.

At each step, the arm with the highest UCB is chosen. In this way, the balance between exploitation and exploration is handled implicitly. Exploitation happens when an arm with high current estimated reward is chosen because of its UCB being the highest among other arms'. Later in time, this UCB may be surpassed by other arms' as these arms have been pull less often, which leads to exploration. UCB-based algorithm outperforms ϵ-greedy algorithms in stationary settings. However, maintaining the UCB estimation requires additional memory and computational recourse.

Continuum-armed bandits: The multi-armed bandit problem we discussed above assumes the reward of each arm to be statistically independent of each other. The expected reward of each arm is, therefore, estimated separately. As a result, the computational and memory cost grows linearly with the number of arms. Fortunately, in many practical settings, the rewards of the arms are often correlated, which allows us to design efficient algorithms when a large number of arms is presented or even when the arms are continuously valued.

Particular to our interest is the work of linearly parameterized bandits [23]. In this model, each arm is represented by a vector $\boldsymbol{a} \in \mathbb{R}^u$ and the reward received from pulling an arm $\boldsymbol{a}$ at time t is assumed to satisfy:

$$r_t(\boldsymbol{a}) = \langle \boldsymbol{a}, \boldsymbol{\theta} \rangle + w_t\,, \tag{1.15}$$

where $\boldsymbol{\theta}$ is an unknown parameter in $\mathbb{R}^u$ and w_t is a zero-mean random variable independent of both t and $\boldsymbol{a}$. Under this assumption, estimating the expected reward of each arm is equivalent to estimating the parameter $\boldsymbol{\theta}$, which can be formulated as a linear regression problem. The estimated parameter $\hat{\boldsymbol{\theta}}_t$ at time t is given by:

$$\hat{\boldsymbol{\theta}}_t = \arg\min_{\boldsymbol{\theta}} \sum_{\tau=1}^{t} \left(r_\tau - \langle \boldsymbol{a}_\tau, \boldsymbol{\theta} \rangle\right)^2 \ . \tag{1.16}$$

The balance between exploitation and exploration can be handled by either the ϵ-greedy algorithm or an UCB-based algorithm. In [23], the uncertainly of the parameter estimation is captured by an "uncertainty ellipsoid". Both the estimated expected reward and the uncertainty ellipsoid contribute to the arm selection at each step.

Contextual bandits: So far in our discussion of the bandit problems, the player selects the arm only based on the historical returns of the arms. We can modify our settings to provide the player some additional information at each step. Now assume that before selecting an arm, the player observes a state variable s, which, when combined with the action variable a, emits an observable feature vector $\boldsymbol{x}(s, a) \in \mathbb{R}^u$. The state variable is further assumed to be independent of the historical action choices. At each time t, the reward r_t depends on the feature vector $\boldsymbol{x}(s_t, a_t)$. In [24], this relationship is assumed to be linear:

$$r_t(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{\theta} \rangle + w_t \ , \tag{1.17}$$

where $\boldsymbol{\theta} \in \mathbb{R}^u$ is an unknown parameter and w_t is a zero-mean random variable independent of both t and $\boldsymbol{x}$. Notice the similarity between the formulation in (1.15) and (1.17). Estimating the expected reward of each arm under each state is equivalent to estimating the parameter $\boldsymbol{\theta}$ which can be formulated as a linear regression problem. The estimated parameter $\hat{\boldsymbol{\theta}}_t$ at time t is given by:

$$\hat{\boldsymbol{\theta}}_t = \arg\min_{\boldsymbol{\theta}} \sum_{\tau=1}^{t} \left(r_\tau - \langle \boldsymbol{x}_\tau, \boldsymbol{\theta} \rangle\right)^2 \ . \tag{1.18}$$

The balance between exploitation and exploration can be handled by either the ϵ-greedy algorithm or an UCB-based algorithm presented in [24]. Although the formulations in (1.15) and (1.17) are very similar, the UCB-based algorithms for each case are significantly different. In linearly parameterized bandits, the reward only depends on the action vector. The exploration can effectively sample the whole reward space without restrictions on the action choices. However, in contextual bandit problems, the feature vector partially depends on the uncontrollable state variable. This state variable hinders the effectiveness of exploring the whole reward space, and the UCB-based algorithm must consider the distribution of the state variable and its effect on the feature vector.

Non-stationary environments: Our discussion has been focusing on the bandit problems in stationary environments. We assume the distribution of each arm's reward does not change with time. However, time-variant environments are common in real-life applications. In order to adapt the arm selection policy to the environmental changes, both the way how expected rewards are estimated and the balance between exploration and exploitation must be modified. In the multi-armed bandit setting, equation (1.13) is rewritten as:

$$\hat{R}_t(a_t) = \hat{R}_{t-1}(a_t) + \alpha_t \left(r_t - \hat{R}_{t-1}(a_t) \right) \ , \tag{1.19}$$

with $0 < \alpha_t \leq 1$. This update scheme is known as stochastic approximation [25]. In a stationary setting, the estimation asymptotically converges to the true expectation when the step sizes satisfy [25]:

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty \ . \tag{1.20}$$

The update step (1.13) can be viewed as a special case of (1.19) with $\alpha_t = \frac{1}{N_t(a_t)}$ which satisfies the convergence conditions (1.20).

In a time-variant environment, we can fix α_t to one value: $\alpha_t = \alpha_0$. Given k reward

samples of arm a, the estimated expected reward can be written as:

$$\hat{R}_k(a) = (1-\alpha_0)^k \hat{R}_0(a) + \alpha_0 \sum_{i=1}^{k} (1-\alpha_0)^{k-i} r_k. \tag{1.21}$$

The earlier an reward is received, the more it is discounted for the estimation. When α_0 is close to 1, the estimation can quickly adapt to the new environment but suffers from a larger variance. When α_0 is close to 0, the estimation has smaller variance but reacts slowly to the environmental changes.

The estimation update step in continuum-armed bandits (1.16) and contextual bandits (1.18) are formulated as least square problems. To modify it for a time-variant environment, we run the regression only on recently collected samples rather than all the samples collected since $t = 1$. The least square problem can be solved efficiently using adaptive filtering techniques such as recursive least square (RLS) and least mean squares (LMS) [26, 27].

In a stationary setting, balancing exploitation and exploration is often based on the assumption that as the number of samples increases, the player becomes more confident about his estimations of the expected rewards. In general, as the trial progresses, the player should gradually reduce the time he spends in exploring, and instead focuses on selecting the arms he believes to return large rewards. However, this assumption is not supported when the reward distribution changes over time. An arm previously believed to return large rewards can become inferior to the others later in the trial. Our confidence on one arm's expected reward estimation depends not only on how many samples we have collected for that arm, but also on how recent these samples are collected. Balancing exploitation and exploration, especially without any knowledge about how the external environment changes over time, becomes very challenging and is often handled by ad hoc techniques [28, 29, 30].

In Chapter 4 we will design an adaptive hardware control algorithm using the bandit frameworks discussed above. The hardware settings correspond to different arms in the bandit framework. When a setting is selected, we measure the hardware's performance

and assign rewards based on some objective function. Optimal performance is achieved when the algorithm learns to choose the setting that returns the highest expected reward. As the environment in which the hardware operates changes overtime, the problem is non-stationary by nature. Two requirements are utterly important for designing adaptive hardware control algorithms. First, the computational and memory complexity must satisfy the constraints posed by the hardware, for that hardware often has limited computational power and memory space. Furthermore, a hardware often needs to perform curtain tasks within a given time frame. The control algorithm needs to finish all the computation within this time constraint. Therefore, simple method such as ϵ-greedy method can be a good choice for balancing exploration and exploitation. The second requirement for an adaptive control algorithm is to successfully adjust to the environmental changes timely. Learning the new optimal arm needs to be efficient. Fortunately, the hardware's performance under different settings are usually correlated. We may also have access to some sensors' data that provides additional information about the hardware or the environment. How these correlated settings and observable information are incorporated into the continuum-armed bandit and the contextual-bandit frameworks to improve the learning efficiency is discussed in Chapter 4.

1.2 Contributions

In Chapter 2, our work is the first to extend the ADMM-based compressive sensing recovery methods to incorporate GANs. Such an extension allows us to exploit the strong priors captured by GANs and significantly increases the recoverable compression ratio. We demonstrate that constructing generative models with structured latent variable plays an important role in fast and stable recovery. Our proposed algorithm achieves comparable performance with a notable speed-up compared to the gradient-based methods. We also provide theoretical guarantees for our algorithm's recovery quality.

In Chapter 3, our work is the first gesture recognition algorithm that directly works

with compressively sensed video data. We also contribute to the time series classification method, allowing clustering and dimension reduction techniques to be applied to time series of different length. The proposed classification method improves both the computational and the memory efficiency. In the algorithm-hardware co-design, we implemented the compression in the sensor front-end and motion parameter estimation in the mixed signal domain. The testing results show significant energy saving over previous work.

In Chapter 4, our work is the first fully adaptive hardware control algorithm for Doherty PA linearization. It is also the first Doherty PA system with bandit/RL-based controls. The learning nature of the bandit-RL frameworks allows our system to adapt to environmental changes and maintain robust performance. We incorporated the properties of the Doherty PA into the algorithm design to achieve high learning efficiency and fast adaption rate.

CHAPTER 2
MACHINE LEARNING IN INVERSE IMAGING: COMPRESSIVE SENSING RECOVERY

In this chapter, we demonstrate how machine learning can be used in solving inverse imaging problems. Imaging hardware with various sensing schemes such as magnetic resonance imaging (MRI) and computed tomography (CT) takes indirect measurements of the object of interest. The goal of the inverse imaging algorithm is to recover the true image from these indirect measurements. Here, we consider the measurements taken by compressive sensing, a technique that allows significant energy saving in data acquisition. As deep learning models have significantly improved the visual quality and accuracy of compressive sensing recovery, we propose an algorithm for signal reconstruction from compressed measurements with image priors captured by a generative model. We search and constrain on latent variable space to make the method stable when the number of compressed measurements is extremely limited. We show that, by exploiting certain structures of the latent variables, the proposed method produces improved reconstruction accuracy and preserves realistic and non-smooth features in the image. Our algorithm achieves high computation speed by projecting between the original signal space and the latent variable space in an alternating fashion.

2.1 Introduction

In compressive sensing (CS), we seek to reconstruct a high-dimensional signal after observing a small number of linearly coded measurements. Mathematically, given a vector $\boldsymbol{x} \in \mathbb{R}^N$, we obtain its linearly compressed representation $\boldsymbol{y}$ in a low dimensional space $\mathbb{R}^M$ $(M \ll N)$ by applying $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}$, where $\boldsymbol{\Phi}$ is the compression matrix in $\mathbb{R}^{M \times N}$. In the case of useful compression, for two distinct signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in the original space, their corresponding representations $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ in the compressed domain also need to be separate.

As $\mathbf{\Phi}$ is an under-determined matrix with a null space, information can not be preserved for all vectors in $\mathbb{R}^N$. Fortunately, real-life signals often have prominent structures and only lie in some subspace of $\mathbb{R}^N$.

Traditional CS focuses on sparse signals. When a signal $\boldsymbol{x}$ has (or approximately has) a k-sparse representation in some basis, the compression matrix $\mathbf{\Phi}$ can be constructed to preserve the distances between two signals in the compressed domain, a property formally known as the restricted isometry property (RIP) [2]. Common methods to construct $\mathbf{\Phi}$ include choosing each entry from an i.i.d. Gaussian [9], a fair Bernoulli, or an independent Sub-Gaussian distribution [10]. Similar results were also established for signals on a smooth manifold. [11]

In order to retrieve the original signal from the compressed measurements, some type of prior knowledge needs to be assumed. When the class of signal is well-studied, the prior knowledge comes from years of experience. For example, the sparsity model is often used to recover compressively sensed natural images. Now with the power of machine learning, we can extract prior knowledge efficiently from collected datasets. The powerful function approximation capability of deep neural networks also allows us to discover and represent more complicated signal structures.

In this chapter, we propose a fast compressive sensing recovery algorithm using generative models with structured latent variables. The prior information of the signals is captured by a generative adversarial network (GAN). The stability of the recovery algorithm is improved when the GAN's latent variable space is well structured. Based on Alternating Direction Methods of Multipliers (ADMM), our algorithm achieves high reconstruction speed by alternatively projecting between the original signal space and the latent variable space, without involving gradient descent.

To the authors' knowledge, our work is the first to extend the ADMM-based CS recovery methods to GANs. Such an extension allows us to exploit the strong priors captured by GANs and significantly increases the recoverable compression ratio. Previous works on CS

recovery using generative model either had limited model capacity [31] or were slow to carry out [18, 32, 33] due to relying heavily on gradient descent. We demonstrate that a structured latent variable space in GAN plays an important role in fast and stable recovery. Our proposed algorithm achieves comparable performance with a notable speed-up compared to the gradient-based methods. Although We present our results by solving compressive sensing recovery problems, our model can be easily generalized to solve other inverse imaging problems.

2.2 Related work

Signal recovery from the compressed measurements can be formulated as an optimization problem of the following form:

$$\min_{\boldsymbol{x}} \; F_{\boldsymbol{y}}(\boldsymbol{x}) + \lambda J(\boldsymbol{x}) \; . \tag{2.1}$$

The first term $F_{\boldsymbol{y}}(\boldsymbol{x})$ is the fidelity term which controls how well a candidate signal $\boldsymbol{x}$ matches the measurement $\boldsymbol{y}$ in the compressed domain. A common choice of $F_{\boldsymbol{y}}(\cdot)$ measures the Euclidean distance in the compressed domain: $||\boldsymbol{y} - \boldsymbol{\Phi x}||_2^2$. The second term $J(\cdot)$ is the property term which encodes the properties the signal of interest must satisfy. In the case of sparse signals, $J(\cdot)$ can be $||\boldsymbol{x}||_0$ or the convex relaxation form $||\boldsymbol{x}||_1$. When a dataset is available, one can "learn" the properties of the signals using machine learning algorithms. Unsupervised learning methods such as Gaussian mixture model (GMM) [31] and variational autoencoder (VAE) [18] have been proposed. A scalar λ controls the trade-off between the fidelity term and the property term.

Recovering sparse signals is a well-studied area. Algorithms such as orthogonal matching pursuit [34], linear programming [2], least angle regression stagewise [35], soft thresholding [36], and approximate message passing [37] were proposed to solve the variations of the optimization problem (2.1).

When $J(\cdot)$ is learned directly from the data, it usually has a non-convex form. Although the global minimum is difficult to find, a local minimum may already result in satisfying results. One can also apply ADMM to (2.1), which leads to iterative solving steps:

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \arg\min_{\boldsymbol{x}} \; F_{\boldsymbol{y}}(\boldsymbol{x}) + \frac{\rho}{2}||\boldsymbol{x} - \boldsymbol{s}^{(k)} + \boldsymbol{\mu}^{(k)}||_2^2 \\
\boldsymbol{s}^{(k+1)} &= \arg\min_{\boldsymbol{s}} \; \lambda J(\boldsymbol{s}) + \frac{\rho}{2}||\boldsymbol{x}^{(k+1)} - \boldsymbol{s} + \boldsymbol{\mu}^{(k)}||_2^2 \\
\boldsymbol{\mu}^{(k+1)} &= \boldsymbol{\mu}^{(k)} + \boldsymbol{x}^{(k+1)} - \boldsymbol{s}^{(k+1)} \; .
\end{aligned} \tag{2.2}$$

ADMM introduces one auxiliary variable $\boldsymbol{s}$ and one dual variable $\boldsymbol{\mu}$. Let $F_{\boldsymbol{y}}(\cdot)$ take the form of $||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}||_2^2$, and the first update step in (2.2) can be easily computed by solving least squares. The second update step is a proximal operator of J, and can be understood as finding a signal $\boldsymbol{s}$ that is close to the target signal $\boldsymbol{x}^{(k+1)} + \boldsymbol{\mu}^{(k)}$ while satisfying the properties encoded in $J(\cdot)$. Such a formulation is often considered as a "denoising" step and inspires many recent works on "plug-and-play" methods. In these works, the second update step is replaced by state-of-the-art denoisers. Instead of explicitly learning $J(\cdot)$ to capture the statistics of the signals, denoisers are trained directly to mimic the behavior of the second update step with the properties of the signals embedded into the denoisers' design. Common denoiser choices include block-matching and 3D filtering (BM3D) [38, 39], and feed-forward neural networks [40, 41]. Adversarial training was proposed in [42, 43] to improve denoising and recovery. Discriminator were used to evaluate the denosing effect, however no generative model was trained to directly capture the statistics of the singal datasets. When a neural network based denoiser is used in the plug-and-play methods, it is often trained using original signals with additive Gaussian noise. Other recovery methods unroll the whole iterative optimization algorithm, such as ADMM [44, 45], projected gradient descent [46, 47, 48], and primal dual hybrid gradient [49], into a neural network and use end-to-end training.

In the aforementioned ADMM framework, $\boldsymbol{x}$ and $\boldsymbol{s}$ are both in the original space $\mathbb{R}^N$

(with the ADMM underlying constraint $\boldsymbol{x} = \boldsymbol{s}$). When generative models are used to capture the signal statistics, latent variables $\boldsymbol{z} \in \mathbb{R}^L$ are usually introduced. [31] used a GMM to model a smooth manifold and searched $\boldsymbol{z}^*$ in the latent space which has a closed form solution. [18] used a VAE to learn a non-linear mapping $G_{gen}(\cdot)$ from $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ to $\boldsymbol{x}$ and applied gradient descent to the optimization problem $\min_{z} ||\boldsymbol{y} - \boldsymbol{\Phi} G_{gen}(\boldsymbol{z})||_2^2 + \lambda ||\boldsymbol{z}||_2^2$. Recent developments in GAN [13, 14] shed new lights on learning signal statistics. [32] used a GAN to capture the image prior and applied gradient descent to the same optimization problem as in [18] for recovery. [33] proposed an algorithm that alternates between one step gradient descent on the fidelity term and searching the latent variable space with the latter still achieved by gradient descent. These gradient based methods suffer from high computational complexity and slow recovery speed. We seek to combine the fast computation from the ADMM-based methods and the strong prior-capture ability from the generative models to achieve fast CS recovery with ultra small number of measurements.

2.3 Algorithms

In the proposed algorithm, we formulate the CS recovery problem as searching $\boldsymbol{x}$ and $\boldsymbol{z}$ in the original signal space and in the latent variable space, respectively:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{z}} \quad & ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}||_2^2 + \lambda H(\boldsymbol{z}) \\ s.t. \quad & \boldsymbol{x} = G_{gen}(\boldsymbol{z}) \ . \end{aligned} \tag{2.3}$$

We denote $G_{gen}(\cdot)$ as the generative model and $H(\cdot)$ as the function that captures the property that the latent variable $\boldsymbol{z}$ should satisfy. Even though the formulation above contains a non-linear equality constraint, we may still solve it by searching a stationary point of its

augmented Lagrangian, which leads to the following ADMM-like update steps:

$$\begin{aligned}
\boldsymbol{x}^{(k+1)} &= \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\mathbf{I}\right)^{-1}\left(\boldsymbol{\Phi}^T\boldsymbol{y} + \rho(G_{gen}(\boldsymbol{z}^{(k)}) - \boldsymbol{\mu}^{(k)})\right) \\
\boldsymbol{z}^{(k+1)} &= \arg\min_{\boldsymbol{z}} \ \lambda H(\boldsymbol{z}) + \frac{\rho}{2}||\boldsymbol{x}^{(k+1)} - G_{gen}(\boldsymbol{z}) + \boldsymbol{\mu}^{(k)}||_2^2 \\
\boldsymbol{\mu}^{(k+1)} &= \boldsymbol{\mu}^{(k)} + \boldsymbol{x}^{(k+1)} - G_{gen}(\boldsymbol{z}^{(k+1)}) \ .
\end{aligned} \tag{2.4}$$

The property imposed by $H(\cdot)$ plays an important role in searching $\boldsymbol{z}$ in the latent variable space. In classic GAN models such as the deep convolutional GAN (DCGAN) proposed in [14], $\boldsymbol{z}$ is assumed to be drawn from a standard multivariate Gaussian distribution. Setting $H(\boldsymbol{z}) = ||\boldsymbol{z}||_2$ is a common practice to enforce this latent space structure. However, this latent variable space is still lack of interpretability and controllability, and how each latent dimension contributes to the generated signal is unclear [15, 16, 50]. The CS recovery performance bound provided in [18] shows the number of required compressed measurements grows linearly with the latent variable dimension.

To enforce better structured latent variable space, we propose training the generative models with an InfoGAN architecture [15], where each latent variable $\boldsymbol{z}$ is split into a codeword $\boldsymbol{c}$ and a "random-noise-like" variable $\boldsymbol{v}$. An InfoGAN is trained to not only minimize the usual GAN's loss function, but also maximize the mutual information between the codeword $\boldsymbol{c}$ and the generated signal $G_{gen}(\boldsymbol{c}, \boldsymbol{v})$. As a results, the codeword $\boldsymbol{c}$ is able to control most of the semantic meaning in the generated signals, while $\boldsymbol{v}$ only adds small variations to the results. A detailed discussion about infoGAN can be found in section 1.1.3 in the introduction chapter. An infoGAN trained on the MNIST dataset is shown in Figure 1.1 and 1.2. The well-structured latent variable space helps CS recovery: When the number of compressed measurements is sufficient, both $\boldsymbol{c}$ and $\boldsymbol{v}$ may be inferred from the measurements, leading to more accurate reconstructions. When the number of compressed measurements is extremely limited, we can recover an approximate signal (within the small variation controlled by $\boldsymbol{v}$) as long as $\boldsymbol{c}$ can be inferred. A theoretical analysis of this recovery

performance is provided in section 2.5.

The x update step in (2.4) solves a least squares problem, while solving for the $\boldsymbol{z}$ update is computationally expensive with gradient descent. Similar to the "plug-and-play" ADMM method, we propose to use a projector neural network G_{proj} to learn the solution of this optimization problem. Notice that during each iteration, $\boldsymbol{x}^{(k+1)}$ contains the noise introduced by the previous least square update. We, therefore, propose to train G_{proj} using randomly sampled latent variables and the noisy version of their generated samples.

Alternatively, we can cascade the projector network and the generator network to form a network $G_{gen}(G_{proj}(\cdot))$ similar to an "autoencoder". We then draw samples directly from the dataset, and train the "autoencoder" to recover these samples from their noisy observations. G_{gen} is fixed during this training process. When this method is used, our proposed algorithm is similar to a "plug-and-play" ADMM model with a generative-model-based denoiser. This similarity may provide a better understanding about the convergence behavior of the proposed algorithm, while our previous derivation provides a better understanding about the importance of using well-structured latent variables. The complete version of the proposed algorithm is summarized in Algorithm 2.1.

2.4 Testing and results

2.4.1 MNIST Dataset

We test the proposed algorithm using the MNIST digits dataset [51]. Selected recovered results are shown in Figure 2.1. We use an i.i.d. Gaussian random matrix $\boldsymbol{\Phi}$ for compression. For comparison purposes, we also include the results of three baseline algorithms. The first algorithm is the "plug-and-play" ADMM with a denoising autoencoder (DAE). As this denoiser is trained directly in the pixel domain (mapping $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$ back to $\boldsymbol{x}$), it fails to capture a strong prior knowledge about the digits, and starts to produce images with large artifacts as the compression rate goes to 32x. The second baseline algorithm uses the well-known total variance (TV) [52] as the regularizer. The third baseline algorithm

Algorithm 2.1 Fast CS recovery using generative models (F-CSRG)

Train a generative model $G_{gen}(\cdot)$ on the dataset.
Method I:
Randomly sample latent variable $\boldsymbol{z}$ following its distribution.
Generate random noise $\boldsymbol{\epsilon}$ according to some distribution.
Construct noisy signal $\tilde{\boldsymbol{x}}$ such that $\tilde{\boldsymbol{x}} = G_{gen}(\boldsymbol{z}) + \boldsymbol{\epsilon}$.
Train a projector network $G_{proj}(\cdot)$ that maps $\tilde{\boldsymbol{x}}$ to $\boldsymbol{z}$.
Method II:
Draw samples of $\boldsymbol{x}$ from the training set.
Generate random noise $\boldsymbol{\epsilon}$ according to some distribution.
Construct noisy signals $\tilde{\boldsymbol{x}}$ such that $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}$.
Train a projector network $G_{proj}(\cdot)$ such that $G_{gen}(G_{proj}(\cdot))$ maps $\tilde{\boldsymbol{x}}$ to $\boldsymbol{x}$.
G_{gen} is fixed during training.

For signal recovery:
Given a compression matrix $\boldsymbol{\Phi}$ and compressed measurements $\boldsymbol{y}$.
while Stopping criteria not met **do**
$\boldsymbol{x}^{(k+1)} = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \rho\mathbf{I}\right)^{-1}\left(\boldsymbol{\Phi}^T\boldsymbol{y} + \rho(G_{gen}(\boldsymbol{z}^{(k)}) - \boldsymbol{\mu}^{(k)})\right)$.
$\boldsymbol{z}^{(k+1)} = G_{proj}(\boldsymbol{x}^{(k+1)} + \boldsymbol{\mu}^{(k)})$.
$\boldsymbol{\mu}^{(k+1)} = \boldsymbol{\mu}^{(k)} + \boldsymbol{x}^{(k+1)} - G_{gen}(\boldsymbol{z}^{(k+1)})$.
end while

uses DCGAN [14] to capture the signal statistics, while the CS recovery is performed using gradient descent as proposed in [18]. We use the same DCGAN's generator as G_{gen} in our F-CSRG algorithm and train a neural network with fully connected layers (784-1024-512-256-100, ReLU activation) as G_{proj}. F-CSRG performs comparably with the gradient-descent-based algorithm, while only takes 1/20 of the computational time. When the number of compressed measurements is significantly reduced, both DCGAN-based models break down due to unstable projections caused by their less structured latent variable space. The best performance is achieved by incorporating InfoGAN (trained as suggested in [15]) into our algorithm, demonstrating the benefit of having well-structured latent variables.

We test three fast recovery algorithms on MNIST digits' testing dataset and measure the Euclidean distance between the true images and the recovered images. The results are shown in Table 2.1. As the compression ratio increases, our algorithm with InfoGAN outperforms the other algorithms. We also use the classification accuracy as another metric

to assess the recovery quality. We train a convolutional neural network as the classifier which achieves 99.3% accuracy on the original MNIST testing dataset. We then apply this MNIST classifier to the images reconstructed by the three algorithms. As shown in Table 2.1, when the number of compressed measurements is extremely limited, our proposed InfoGAN algorithm achieves significantly higher accuracy than the other two methods, as a result of the InfoGAN's structured latent variable space.

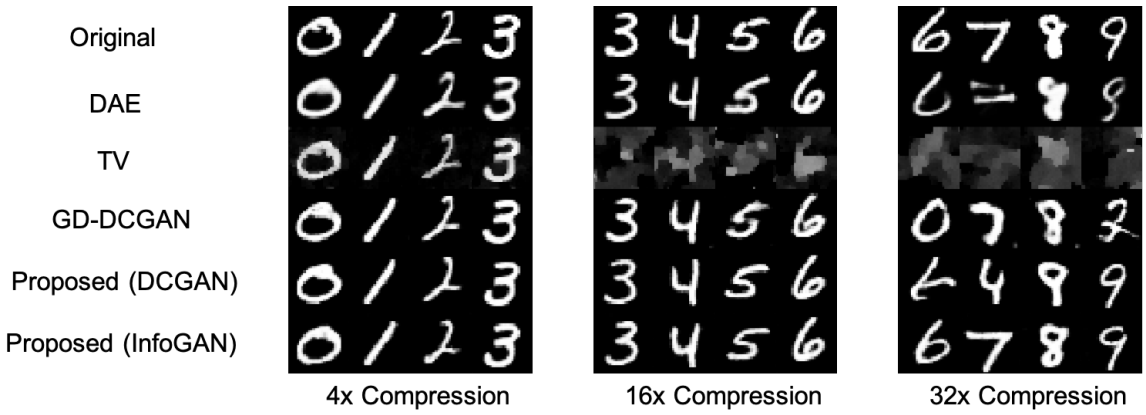


Figure 2.1: Comparison of selected MNIST digits recovered by different algorithms.

Compression Ratio	DAE	F-CSRG with DCGAN	F-CSRG with InfoGAN
4x	2.20 / 98.2%	2.25 / 98.3%	2.68 / 97.7%
8x	2.54 / 97.8%	2.72 / 97.3%	3.06 / 97.2%
16x	3.23 / 94.8%	3.70 / 91.7%	3.79 / 93.8%
32x	5.13 / 73.5%	5.86 / 66.4%	5.37 / 77.4%
64x	7.33 / 41.8%	7.91 / 36.2%	7.43 / 48.0%

Table 2.1: Average reconstruction error (measurened as the Euclidean distance) and classification accuracy of the reconstructed digits on MNIST digits' testing dataset

2.4.2 Celeb A Datasets

We also test the proposed algorithm using the CelebA dataset [53]. Each image is of dimension 32×32, cropped and downsampled from the original dataset. Generative models are trained based on the standard DCGAN and InfoGAN architectures as proposed in the

Figure 2.2: Comparison of selected CelebA images recovered by different algorithms

original papers. For InfoGAN, we use 5 categorical codes (one-hot encoding with 10 classes), 5 continuous codes, and a randomness variable of length 128, producing a latent variable of length 183. We train a fully connected network (1024-512-256-183, ReLu activation) as G_{proj}. Bases on whether the desired codeword is categorical or continuous, we use softmax as the activation function or simply skip activation on the output layer. We test the proposed algorithm with 4x, 8x, and 16x compression. For comparison purposes, we also include the results of two baseline algorithms. Similar to the testing on the MNIST dataset, the first baseline algorithm uses the TV regularization, and the second one uses the "plug-and-play" ADMM with a DAE trained directly in the pixel domain. Selected recovered results are shown in Figure 2.2.

As the compression rate increases, the quality of the images recovered by TV regularization degrades and few features on the face can be recovered. The recovery algorithm using a DAE produces comparable or sometimes even better reconstruction in the case of 4x and 8x compression, but becomes unstable under extremely high compression rate. In contrast, the proposed F-CSRG method is very stable against the drop in the number of compressed measurements. In addition, because generative model provides a strong image prior that assumes face images to have sharp features and not necessarily smooth everywhere, images reconstructed by the proposed algorithm preserve high-frequency contents of the original images. Out testing results also show that F-CSRG works better with an InfoGAN implementation than with a DCGAN. As we have discussed in previous sections,

this improvement in recovered image quality comes from the more structured latent variable space produced by the InfoGAN's codeword [1].

2.5 Theoretical recovery guarantee

We begin this section by stating the main theoretical result that provides a guarantee on the recovery performance in the compressed domain with respect to the performance without compression.

Theorem 2.5.1 *Assume a Lipschitz continuous mapping $G(\cdot,\cdot) : \mathbb{R}^D \times \mathbb{R}^{L-D} \to \mathbb{R}^N$ that has Lipschitz constant T and satisfies: $||G(\boldsymbol{c},\boldsymbol{v}_1) - G(\boldsymbol{c},\boldsymbol{v}_2)|| \leq \beta,\ \forall \boldsymbol{c} \in \mathcal{B}^D(r_c), \boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{B}^{L-D}(r_v)$, where $\mathcal{B}^L(r)$ denotes a norm ball in $\mathbb{R}^L$ with radius r: $\mathcal{B}^L(r) = \{z | z \in \mathbb{R}^L, ||z|| < r\}$ and $||\cdot||$ denotes the L_2 norm. Assume also a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$. The compressed measurements are obtained as $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + w$, where $\boldsymbol{x}^*$ is the desired signal and w is an independent additive noise. Define $\bar{\boldsymbol{x}} = \underset{\boldsymbol{x}\in G(\mathcal{B}^D(r_c),\mathcal{B}^{L-D}(r_v))}{\arg\min} ||\boldsymbol{x}^* - \boldsymbol{x}||$ and $\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in G(\mathcal{B}^D(r_c),\mathcal{B}^{L-D}(r_v))}{\arg\min} ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}||$.*

For some $0 < \epsilon < \frac{1}{2}$, $0 < \delta < 1$, if $M \geq \mathcal{O}\left(\frac{D}{\epsilon^2}\log\frac{Tr_c}{\delta}\right)$, then with probability at least $1-\delta$:

$$||\boldsymbol{x}^* - \hat{\boldsymbol{x}}|| \leq \frac{5 + 2\sqrt{\frac{N}{M}} - \epsilon}{1-\epsilon}||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| + \frac{6 + 2\sqrt{\frac{N}{M}} - 2\epsilon}{1-\epsilon}\beta + \frac{2}{1-\epsilon}||w|| + \mathcal{O}(\delta)\ . \quad (2.5)$$

Furthermore, if $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2}\log\frac{T\sqrt{r_c^2+r_v^2}}{\delta}\right)$, then with probability at least $1-\delta$:

$$||\boldsymbol{x}^* - \hat{\boldsymbol{x}}|| \leq \frac{5 + 2\sqrt{\frac{N}{M}} - \epsilon}{1-\epsilon}||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| + \frac{2}{1-\epsilon}||w|| + \mathcal{O}(\delta)\ . \quad (2.6)$$

The first requirement of this theorem is a Lipschitz continuous mapping $G(\cdot)$ from $\mathbb{R}^L$ to $\mathbb{R}^n$. Lipschitz continuity requires the difference between two outputs of the mapping

[1] A Tensorflow implementation can be found at https://github.com/sihan-zeng/f-csrg.

generated by two inputs with bounded difference also be bounded up to a constant factor T:

$$||G(\boldsymbol{z}_1) - G(\boldsymbol{z}_2)|| \leq T||\boldsymbol{z}_1 - \boldsymbol{z}_2|| \,. \tag{2.7}$$

We also require the input variable of the mapping to be divided into two variables each with a bounded norm. Their contribution to the output of the mapping are significantly different. The first vector $\boldsymbol{c}$ controls the most variations, and once it is fixed, the other input vector $\boldsymbol{v}$'s contribution to the output can be bounded by a small constant β. These requirements can be satisfied by well-trained InfoGAN naturally. When sampling from a bounded latent variable space, the outputs of a generative model fall on the manifold described by the training data. The input of an InfoGAN is also separated into a codeword and a randomness variable. The former controls the major variations in the generated results while the latter only adds some details.

The compressive sensing matrix $\boldsymbol{\Phi}$ is a random Gaussian matrix, whose entries are i.i.d. sampled. Our theoretical results can be extended to other types of random matrices by plugging in their corresponding concentration bounds. The compressed measurements are assumed to contain some independent additive noise w. We confine our search of the true solution in the output space of the generative model. Without compression, we define $\bar{\boldsymbol{x}}$ as the true signal $\boldsymbol{x}^*$'s projection onto the generative model's output space. $\bar{\boldsymbol{x}}$ has the shortest euclidean distance to $\boldsymbol{x}^*$ compared to all other points on the generative model's output space. This distance is unlikely to be completely zero due to the generative model's imperfect matching to the true signal space. This fundamental constraint present in the generative model itself without compressive sensing's involvement poses a lower bound on the recovery performance. With compressive sensing, we only have access to the compressed measurements $\boldsymbol{y}$. In this case, we still conduct our search in the generative model's output space, trying to find the signal that, after compression, best matches y. Denote this signal as $\hat{\boldsymbol{x}}$ and we seek to bound how far compressive sensing pushes this recovered signal away

from the true signal $\boldsymbol{x}^*$ with respect to the distance between $\bar{\boldsymbol{x}}$ and $\boldsymbol{x}^*$.

The performance bound we provide above contains two parts. In the first part, we consider the case when the number of compressed measurements are very limited. The required number of measurements in this case has a linear relationship with the dimension of the codeword space D. In section 2.4, we have shown that D is often extremely small. The effect of compression mainly shows in two terms in the right hand side of (2.5). The error caused by the generative model's approximation of the true signal's distribution is magnified by a factor of $\sqrt{\frac{N}{M}}$. Moreover, some details in the original signal are inevitably lost due to the small number of measurements. These details, bounded by β, is also magnified by the order of $\sqrt{\frac{N}{M}}$ and contributes to the recovery error bound. However as the number of measurements increases, the details in the original signal become more likely to be recovered. In the second part of the theorem, when the number of measurements satisfies some linear relationship with the dimension of the whole latent variable space L, the term involving β is dropped from the right hand side in (2.6). This bound has been previously derived in [18]. We extend the previous work to generative models whose latent variable space is well-structured and provided recovery guarantees adaptive to the number of measurements.

2.5.1 Proof of the main theorem

Like many previous theoretical works in compressive sensing, our proof starts from the Johnson-Lindenstrauss (JL) lemma, which states the length preserving property of applying random projection on a finite set of vectors.

Lemma 2.5.2 (JL lemma with i.i.d. Gaussian matrix) *For a finite set $\mathcal{S}$ of Q vectors ($|\mathcal{S}| = Q$) in $\mathbb{R}^N$, a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M\times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$, and some $0 < \epsilon < \frac{1}{2}$, we have with probability at least $1 - 2Qe^{-\frac{\epsilon^2 M}{8}}$, $\forall \boldsymbol{x} \in \mathcal{S}$:*

$$(1-\epsilon)||\boldsymbol{x}|| \leq ||\boldsymbol{\Phi}\boldsymbol{x}|| \leq (1+\epsilon)||\boldsymbol{x}|| \,. \tag{2.8}$$

In other words, for some $0 < \epsilon < \frac{1}{2}$ and probability $0 < \delta < 1$, $(1-\epsilon)||\boldsymbol{x}|| \leq ||\boldsymbol{\Phi}\boldsymbol{x}|| \leq$

$(1+\epsilon)||\boldsymbol{x}||$ *holds with probability at least* $1-\delta$ *as long as:*

$$M \geq \frac{8}{\epsilon^2}\log\frac{2Q}{\delta}\ . \tag{2.9}$$

Pairwise distance preserving in a covering set: Now consider a Lipschitz mapping $G(\cdot):\mathbb{R}^L \to \mathbb{R}^N$ with Lipschitz constant T, and a norm ball $\mathcal{B}^L(r)=\{z|z\in\mathbb{R}^L, ||z||<r\}$. We can construct an $\frac{\eta}{T}$-covering for $\mathcal{B}^L(r)$, denoted as $\mathcal{C}^L(\frac{\eta}{T})$. Then by the definition and covering and Lipschitz continuity $\mathcal{C}^N(\eta) = G(\mathcal{C}^L(\frac{\eta}{T}))$ is an η-covering for $G(\mathcal{B}^L(r))$. Applying the classic bound on the covering number, we have:

$$|\mathcal{C}^N(\eta)| = |\mathcal{C}^L(\frac{\eta}{T})| \leq (\frac{3Tr}{\eta})^L\ . \tag{2.10}$$

To prove the pairwise distance preserving property, we construct a new set $\mathcal{D}^N(\eta)=\{\boldsymbol{d}|\boldsymbol{d}=\boldsymbol{x}_1-\boldsymbol{x}_2,\ \boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{C}^N(\eta)\}$. Then by the way we construct $\mathcal{D}^N(\eta)$, we have:

$$|\mathcal{D}^N(\eta)| = \binom{|\mathcal{C}^N(\eta)|}{2} \leq \frac{1}{2}|\mathcal{C}^N(\eta)|^2 \leq \frac{1}{2}(\frac{3Tr}{\eta})^{2L}\ . \tag{2.11}$$

Plugging the upper bound as Q in the JL Lemma, we derive that as long as

$$M \geq \frac{8}{\epsilon^2}\log(\frac{1}{\delta}(\frac{3Tr}{\eta})^{2L})\ , \tag{2.12}$$

$(1-\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| < ||\boldsymbol{\Phi}(\boldsymbol{x}_1-\boldsymbol{x}_2)|| < (1+\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2||$ holds with probability at least $1-\delta$, for all $\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{C}^N(\eta)$. And by setting $\eta = 3\delta^{\frac{2L-1}{2L}} = \mathcal{O}(\delta)$, we have:

$$M \geq \frac{16L}{\epsilon^2}\log\frac{Tr}{\delta}\ , \tag{2.13}$$

is sufficient to have $(1-\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| < ||\boldsymbol{\Phi}(\boldsymbol{x}_1-\boldsymbol{x}_2)|| < (1+\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2||$ hold with probability at least $1-\delta$, for all $\boldsymbol{x}_1,\boldsymbol{x}_2\in\mathcal{C}^N(\mathcal{O}(\delta))$.

RIP for all signals in the output space of a generative model: To generalize the dis-

tance preserving property to all $\boldsymbol{x}_1, \boldsymbol{x}_2 \in G(\mathcal{B}^L(r))$, we apply a technique named "chaining". First, we construct a chain of coverings for $G(\mathcal{B}^L(r))$ with increasing radius. Then for every $\boldsymbol{x} \in G(\mathcal{B}^L(r))$, we can find a chain of vectors $\boldsymbol{q}_F \in \mathcal{C}_F^N(\frac{\eta}{2^F})$, $\boldsymbol{q}_{F-1} \in \mathcal{C}_{F-1}^N(\frac{\eta}{2^{F-1}})$, ..., $\boldsymbol{q}_0 \in \mathcal{C}_0^N(\eta)$ that satisfies $||\boldsymbol{x} - \boldsymbol{q}_F|| \leq \frac{\eta}{2^F}$ and $||\boldsymbol{q}_{f+1} - \boldsymbol{q}_f|| \leq \frac{\eta}{2^f}, \forall f = 0, 1, \ldots, F-1$. The construction starts from finding $\boldsymbol{q}_F$ and then moves down along the chain. By the definition of covering, $\boldsymbol{q}_f$ must exists given $\boldsymbol{q}_{f+1}$. Even though the chaining vectors change with $\boldsymbol{x}$, the coverings are fixed. In this way, the bound we derive later will hold for all $\boldsymbol{x}$ simultaneously. The chaining processing also implies:

$$\begin{aligned} ||\boldsymbol{x} - \boldsymbol{q}_0|| &= ||(\boldsymbol{x} - \boldsymbol{q}_F) + (\boldsymbol{q}_F - \boldsymbol{q}_{F-1}) + \ldots + (\boldsymbol{q}_1 - \boldsymbol{q}_0)|| \\ &\leq ||\boldsymbol{x} - \boldsymbol{q}_F|| + ||\boldsymbol{q}_F - \boldsymbol{q}_{F-1}|| + \ldots + ||\boldsymbol{q}_1 - \boldsymbol{q}_0|| \\ &\leq \sum_{f=0}^{F} \frac{\eta}{2^f} \leq 2\eta \, . \end{aligned} \tag{2.14}$$

With the constructed chain, the compressed distance from $\boldsymbol{x}$ to a point in $\mathcal{C}_0^N(\eta)$ covering can be bounded as:

$$\begin{aligned} ||\boldsymbol{\Phi}(\boldsymbol{x} - \boldsymbol{q}_0)|| &= ||\boldsymbol{\Phi}(\boldsymbol{x} - \boldsymbol{q}_F) + \boldsymbol{\Phi}(\boldsymbol{q}_F - \boldsymbol{q}_{F-1}) + \ldots + \boldsymbol{\Phi}(\boldsymbol{q}_1 - \boldsymbol{q}_0)|| \\ &\leq ||\boldsymbol{\Phi}(\boldsymbol{x} - \boldsymbol{q}_F)|| + ||\boldsymbol{\Phi}(\boldsymbol{q}_F - \boldsymbol{q}_{F-1})|| + \ldots + ||\boldsymbol{\Phi}(\boldsymbol{q}_1 - \boldsymbol{q}_0)|| \, . \end{aligned} \tag{2.15}$$

Construct new sets $\mathcal{D}_f^N = \{\boldsymbol{d} | \boldsymbol{d} = \boldsymbol{q} - \boldsymbol{q}', \ \boldsymbol{q} \in \mathcal{C}_{f+1}^N(\frac{\eta}{2^{f+1}}), \ \boldsymbol{q}' \in \mathcal{C}_f^N(\frac{\eta}{2^f})\}$. Then by the way we construct $\mathcal{D}_f^N$, we have:

$$|\mathcal{D}^N| \leq |\mathcal{C}_{f+1}^N(\frac{\eta}{2^{f+1}})| \, |\mathcal{C}_{f+1}^N(\frac{\eta}{2^f})| \leq |\mathcal{C}_{f+1}^N(\frac{\eta}{2^{f+1}})|^2 \leq (\frac{6Tr}{\eta})^{2L} 4^{fL} \, . \tag{2.16}$$

To bound each term in (2.15), we apply a variation of the JL lemma:

Lemma 2.5.3 *For a finite set $\mathcal{S}$ of Q vectors ($|\mathcal{S}| = Q$) in $\mathbb{R}^N$, a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$, and some $\epsilon > 0$, we have*

with probability at least $1 - Q((1+\epsilon)e^{-\epsilon})^{M/2}$, $\forall \boldsymbol{x} \in \mathcal{S}$*:*

$$||\boldsymbol{\Phi}\boldsymbol{x}|| \leq (1+\epsilon)||\boldsymbol{x}|| \,. \tag{2.17}$$

Plugging (2.16) as Q in Lemma 2.5.3, we derive that with probability at least $1 - \delta_f = 1 - (\frac{6Tr}{\eta})^{2L} 4^{fL} ((1+\epsilon_f)e^{-\epsilon_f})^{\frac{M}{2}}$,

$$||\boldsymbol{\Phi}(\boldsymbol{q}_{f+1} - \boldsymbol{q}_f)|| \leq (1+\epsilon_f)||\boldsymbol{q}_{f+1} - \boldsymbol{q}_f|| \leq (1+\epsilon_f)\frac{\eta}{2^f} \,. \tag{2.18}$$

By applying a union bound, we have with probability at least $1 - \sum_{f=0}^{F-1} \delta_f$:

$$\sum_{f=0}^{F-1} ||\boldsymbol{\Phi}(\boldsymbol{q}_{f+1} - \boldsymbol{q}_f)|| \leq \sum_{f=0}^{F-1} (1+\epsilon_f)\frac{\eta}{2^f} \,. \tag{2.19}$$

In order for this bound to be useful, we require that $\sum_{f=0}^{F-1} \delta_f$ and $\sum_{f=0}^{F-1} (1+\epsilon_f)\frac{\eta}{2^f}$ to both converge as $F \to \infty$. Let's consider the following sequence of ϵ_f:

$$\epsilon_f = \begin{cases} \epsilon_0(1+f), & f \geq \left(\frac{1}{2\epsilon_0}\right)^2 - 1 \\ \epsilon_0\sqrt{1+f}, & f < \left(\frac{1}{2\epsilon_0}\right)^2 - 1 \,, \end{cases} \tag{2.20}$$

with some $\epsilon_0 < \frac{1}{2}$. Denote the splitting threshold as $\bar{f}$: $\bar{f} = \left(\frac{1}{2\epsilon_0}\right)^2 - 1$. Then (2.19) is in the order of η:

$$\begin{aligned} \sum_{f=0}^{\infty} (1+\epsilon_f)\frac{\eta}{2^f} &= \sum_{f=0}^{\bar{f}} (1+\epsilon_0\sqrt{1+f})\frac{\eta}{2^f} + \sum_{f=\bar{f}}^{\infty} (1+\epsilon_0(1+f))\frac{\eta}{2^f} \\ &\leq \sum_{f=0}^{\infty} (1+\epsilon_0(1+f))\frac{\eta}{2^f} \\ &= (2+4\epsilon_0)\eta \,. \end{aligned} \tag{2.21}$$

To bound the probability term $1 - \sum_f \delta_f$, we have:

$$\sum_{f=0}^{\infty} \delta_f = \sum_{f=0}^{\infty} (\frac{6Tr}{\eta})^{2L} 4^{fL} ((1+\epsilon_f) e^{-\epsilon_f})^{\frac{M}{2}} \ . \tag{2.22}$$

Notice when $\epsilon_f < \frac{1}{2}$, $(1+\epsilon_f)e^{-\epsilon_f} < e^{-\frac{1}{4}\epsilon_f^2}$, and when $\epsilon_f \geq \frac{1}{2}$, $(1+\epsilon_f)e^{-\epsilon_f} > e^{-\frac{1}{14}\epsilon_f}$. In our proposed ϵ sequence (2.20), $\epsilon_f \geq \frac{1}{2}$ if and only if $f \geq \bar{f}$. Therefore, we bound the probability term by:

$$\begin{aligned}
\sum_{f=0}^{\infty} \delta_f &= (\frac{6Tr}{\eta})^{2L} \left(\sum_{f=0}^{\bar{f}} 4^{fL} ((1+\epsilon_f) e^{-\epsilon_f})^{\frac{M}{2}} + \sum_{f=\bar{f}}^{\infty} 4^{fL} ((1+\epsilon_f) e^{-\epsilon_f})^{\frac{M}{2}} \right) \\
&< (\frac{6Tr}{\eta})^{2L} \left(\sum_{f=0}^{\bar{f}} 4^{fL} e^{-\frac{M}{8}\epsilon_f^2} + \sum_{f=\bar{f}}^{\infty} 4^{fL} e^{-\frac{M}{14}\epsilon_f} \right) \\
&= (\frac{6Tr}{\eta})^{2L} \left(e^{-\frac{M}{8}\epsilon_0^2} \sum_{f=0}^{\bar{f}} \left(4^L e^{-\frac{M}{8}\epsilon_0^2} \right)^f + e^{-\frac{M}{14}\epsilon_0} \sum_{f=\bar{f}}^{\infty} \left(4^L e^{-\frac{M}{14}\epsilon_0} \right)^f \right) \ .
\end{aligned} \tag{2.23}$$

Given $\epsilon_0 < \frac{1}{2}$, we have $e^{-\frac{M}{14}\epsilon_0} < e^{-\frac{M}{8}\epsilon_0^2}$. By denoting δ as the upper bound of the sum of δ_f, we have:

$$\sum_{f=0}^{\infty} \delta_f < (\frac{6Tr}{\eta})^{2L} e^{-\frac{M}{8}\epsilon_0^2} \sum_{f=0}^{\infty} \left(4^L e^{-\frac{M}{8}\epsilon_0^2} \right)^f \leq \delta \ . \tag{2.24}$$

Then by setting $\eta = \delta^{\frac{2L-1}{2L}} = \mathcal{O}(\delta)$, the inequality above leads to:

$$M \geq \frac{8}{\epsilon_0^2} \log \left((\frac{6Tr}{\delta})^{2L} + 2^{2L} \right) \ . \tag{2.25}$$

Since $||\boldsymbol{\Phi}(\boldsymbol{x} - \boldsymbol{q}_F)|| \to 0$ when $F \to \infty$, we conclude that with $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2} \log \frac{Tr}{\delta}\right)$, and probability at least $1 - \delta$, for any $\boldsymbol{x} \in G(\mathcal{B}^k(r)),\ \exists \boldsymbol{q} \in \mathcal{C}^N(\mathcal{O}(\delta))$, such at

$$||\boldsymbol{\Phi}(\boldsymbol{x} - \boldsymbol{q})|| \leq \mathcal{O}(\delta) \ . \tag{2.26}$$

Now, for every pair of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in G(\mathcal{B}^L(r))$, we apply chaining to find two

corresponding vectors $\boldsymbol{q}_1, \boldsymbol{q}_2$ such that $\boldsymbol{q}_1, \boldsymbol{q}_2 \in \mathcal{C}^N(\mathcal{O}(\delta))$, and according to (2.14), $||\boldsymbol{x}_1 - \boldsymbol{q}_1|| \leq \mathcal{O}(\delta)$, $||\boldsymbol{x}_2 - \boldsymbol{q}_2|| \leq \mathcal{O}(\delta)$. We then apply the triangular inequality:

$$||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_2)|| \leq ||\boldsymbol{\Phi}(\boldsymbol{q}_1 - \boldsymbol{q}_2)|| + ||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{q}_1)|| + ||\boldsymbol{\Phi}(\boldsymbol{q}_2 - \boldsymbol{x}_2)|| \,. \tag{2.27}$$

The first term on the right hand side is the distance between two vectors on the covering set. We have shown that with $M \geq \frac{16L}{\epsilon^2} \log \frac{Tr}{\delta}$ and probability at least $1 - \delta$, we have:

$$\begin{aligned} ||\boldsymbol{\Phi}(\boldsymbol{q}_1 - \boldsymbol{q}_2)|| &\leq (1+\epsilon)||\boldsymbol{q}_1 - \boldsymbol{q}_2|| = (1+\epsilon)||\boldsymbol{q}_1 - \boldsymbol{q}_2 - \boldsymbol{x}_1 + \boldsymbol{x}_1 - \boldsymbol{x}_2 + \boldsymbol{x}_2|| \\ &\leq (1+\epsilon)||\boldsymbol{x}_1 - \boldsymbol{x}_2|| + (1+\epsilon)||\boldsymbol{x}_1 - \boldsymbol{q}_1|| + (1+\epsilon)||\boldsymbol{x}_2 - \boldsymbol{q}_2|| \\ &\leq (1+\epsilon)||\boldsymbol{x}_1 - \boldsymbol{x}_2|| + (1+\epsilon)\mathcal{O}(\delta) \;. \end{aligned} \tag{2.28}$$

We can now obtain the pairwise distance preserving property by plugging (2.28) and (2.26) into (2.27) and apply the union bound on the probabilities. We have only shown the derivation of the upper bound. The lower bound can be derived using similar techniques. The distance preserving property is formally stated in Lemma 2.5.4.

Lemma 2.5.4 *Assume a Lipschitz continuous mapping $G(\cdot) : \mathbb{R}^L \to \mathbb{R}^N$ with Lipschitz constant T and a norm ball $\mathcal{B}^L(r) = \{z | z \in \mathbb{R}^L, ||z|| < r\}$ where $||\cdot||$ denotes the L_2 norm. Assume also a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$. Given some $0 < \epsilon < \frac{1}{2}$, $0 < \delta < 1$, if $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2} \log \frac{Tr}{\delta}\right)$, then with probability at least $1 - \delta$:*

$$(1-\epsilon)||\boldsymbol{x}_1 - \boldsymbol{x}_2|| - \mathcal{O}(\delta) \leq ||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_2)|| \leq (1+\epsilon)||\boldsymbol{x}_1 - \boldsymbol{x}_2|| + \mathcal{O}(\delta) \tag{2.29}$$

holds $\forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in G(\mathcal{B}^L(r))$.

Compressive sensing recovery guarantee using a generative model: When the compressed measurements $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + w$ are collected, we search in the subspace generated by $G(\mathcal{B}^L(r))$ rather than $\mathbb{R}^n$. Define $\bar{\boldsymbol{x}}$ as the true signal's projection onto the output space of

the generative model when no compression is presented:

$$\bar{\boldsymbol{x}} = \underset{\boldsymbol{x} \in G(\mathcal{B}^L(r))}{\arg\min} ||\boldsymbol{x}^* - \boldsymbol{x}|| \,. \tag{2.30}$$

When given only the compressed measurements $\boldsymbol{y}$, we search the generative model's output space to find a signal that, after compression, best matches the compressed observations. Define this solution as $\hat{\boldsymbol{x}}$:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in G(\mathcal{B}^L(r))}{\arg\min} ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}|| \,. \tag{2.31}$$

To show the guarantee of the compressive sensing recovery is to show that $||\boldsymbol{x}^* - \hat{\boldsymbol{x}}||$ is comparable to $||\boldsymbol{x}^* - \bar{\boldsymbol{x}}||$, essentially demonstrating that the compression has very small influence on the recovery. We start the proof by applying the triangular inequality:

$$||\boldsymbol{x}^* - \hat{\boldsymbol{x}}|| \leq ||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| + ||\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}}|| \,. \tag{2.32}$$

Since $\hat{\boldsymbol{x}}, \bar{\boldsymbol{x}} \in G(\mathcal{B}^k(r))$, Lemma 2.5.4 indicates that with $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2} \log \frac{Tr}{\delta}\right)$ and probability at least $1 - \delta$:

$$\begin{aligned} ||\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}}|| &\leq \frac{||\boldsymbol{\Phi}(\bar{\boldsymbol{x}} - \hat{\boldsymbol{x}})|| + \mathcal{O}(\delta)}{1 - \epsilon} \\ &\leq \frac{||\boldsymbol{y} - \boldsymbol{\Phi}\bar{\boldsymbol{x}}|| + ||\boldsymbol{y} - \boldsymbol{\Phi}\hat{\boldsymbol{x}}|| + \mathcal{O}(\delta)}{1 - \epsilon} \\ &\leq \frac{2||\boldsymbol{y} - \boldsymbol{\Phi}\bar{\boldsymbol{x}}|| + \mathcal{O}(\delta)}{1 - \epsilon} \\ &\leq \frac{2||\boldsymbol{\Phi}(\boldsymbol{x}^* - \bar{\boldsymbol{x}})|| + 2||w|| + \mathcal{O}(\delta)}{1 - \epsilon} \,. \end{aligned} \tag{2.33}$$

To bound the term $||\boldsymbol{\Phi}(\boldsymbol{x}^* - \bar{\boldsymbol{x}})||$, we apply the results of the random matrix [54, 55, 56]:

Lemma 2.5.5 *For a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ whose each entry is sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$, we have with probability at least $1 - 2e^{-\frac{M}{2}}$, $\forall \boldsymbol{x} \in \mathbb{R}^N$:*

$$||\boldsymbol{\Phi}\boldsymbol{x}|| \leq (2 + \sqrt{\frac{N}{M}})||\boldsymbol{x}|| \,. \tag{2.34}$$

That is, with probability at least $1 - \delta' = 1 - 2e^{-\frac{M}{2}}$:

$$||\mathbf{\Phi}(\boldsymbol{x}^* - \bar{\boldsymbol{x}})|| \leq (2 + \sqrt{\frac{N}{M}})||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| \ . \tag{2.35}$$

It can be shown that with $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2}\log\frac{Tr}{\delta}\right)$, a small ϵ, and some reasonable values for r, T, L, $\delta' < \delta$. By plugging (2.35) into (2.33) and combining the two probability with a union bound, we arrive at the compressive sensing recovery guarantee using a generative model:

Lemma 2.5.6 *Assume a Lipschitz continuous mapping $G(\cdot) : \mathbb{R}^L \to \mathbb{R}^N$ with Lipschitz constant T and a norm ball $\mathcal{B}^L(r) = \{z | z \in \mathbb{R}^L, ||z|| < r\}$ where $||\cdot||$ denotes the L_2 norm. Assume also a random matrix $\mathbf{\Phi} \in \mathbb{R}^{M\times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$. The compressed measurements are obtained as $\boldsymbol{y} = \mathbf{\Phi}\boldsymbol{x}^* + w$, where $\boldsymbol{x}^*$ is the desired signal and w is an independent additive noise. Define $\bar{\boldsymbol{x}} = \underset{\boldsymbol{x}\in G(\mathcal{B}^L(\boldsymbol{z}))}{\arg\min} ||\boldsymbol{x}^* - \boldsymbol{x}||$ and $\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}\in G(\mathcal{B}^L(\boldsymbol{z}))}{\arg\min} ||\boldsymbol{y} - \mathbf{\Phi}\boldsymbol{x}||$.*

If $M \geq \mathcal{O}\left(\frac{L}{\epsilon^2}\log\frac{Tr}{\delta}\right)$, then with probability at least $1 - \delta$:

$$||\boldsymbol{x}^* - \hat{\boldsymbol{x}}|| \leq \frac{5 + 2\sqrt{\frac{N}{M}} - \epsilon}{1 - \epsilon}||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| + \frac{2}{1 - \epsilon}||w|| + \mathcal{O}\left(\delta\right) \ . \tag{2.36}$$

Compressive sensing recovery guarantee using a generative model with structured latent variable space: We now impose some structures in the input space of the generative model. Assume a Lipschitz continuous mapping $G(\cdot, \cdot) : \mathbb{R}^D \times \mathbb{R}^{L-D} \to \mathbb{R}^N$ with Lipschitz constant T further satisfies: $||G(\boldsymbol{c}, \boldsymbol{v}_1) - G(\boldsymbol{c}, \boldsymbol{v}_2)|| \leq \beta, \ \forall \boldsymbol{c} \in \mathcal{B}^D(r_c), \boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{B}^{L-D}(r_v)$. The input space of the generative model is divided into two subspaces. The first input vector $\boldsymbol{c} \in \mathbb{R}^D$ controls the most variations, and once it is fixed the magnitude of the other input vector $\boldsymbol{v} \in R^{L-D}$'s contribution to the output is bounded by a small constant β.

As in the previous proof, we start from the step of constructing a covering set $\mathcal{C}^N(\eta)$ for $G(\mathcal{B}(r_c), \boldsymbol{v}_0)$, where $\boldsymbol{v}_0$ is arbitrary and fixed. As the generated vectors are fully controlled

by $\boldsymbol{c}$ in this way, we have:

$$|\mathcal{C}^n(\eta)| \leq (\frac{3Tr_c}{\eta})^D \ . \tag{2.37}$$

By the assumption of the generative model, for any two signals generated by the model $\boldsymbol{x}_1 = G(\boldsymbol{c}_1, \boldsymbol{v}_1)$, $\boldsymbol{x}_2 = G(\boldsymbol{c}_2, \boldsymbol{v}_2)$, we can find $\boldsymbol{x}_{c1} = G(\boldsymbol{c}_1, \boldsymbol{v}_0)$ and $\boldsymbol{x}_{c2} = G(\boldsymbol{c}_2, \boldsymbol{v}_0)$ such that:

$$||\boldsymbol{x}_1 - \boldsymbol{x}_{c1}|| \leq \beta, \ ||\boldsymbol{x}_2 - \boldsymbol{x}_{c2}|| \leq \beta \ . \tag{2.38}$$

Applying the triangular inequality, the pairwise distance after compression can be bounded by:

$$||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_2)|| \leq ||\boldsymbol{\Phi}(\boldsymbol{x}_{c1} - \boldsymbol{x}_{c2})|| + ||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_{c1})|| + ||\boldsymbol{\Phi}(\boldsymbol{x}_2 - \boldsymbol{x}_{c2})|| \ . \tag{2.39}$$

As $\boldsymbol{x}_{c1}, \boldsymbol{x}_{c2} \in G(\mathcal{B}(r_c), \boldsymbol{v}_0)$, we apply Lemma 2.5.4. With $M \geq \mathcal{O}\left(\frac{D}{\epsilon^2} \log \frac{Tr_c}{\delta}\right)$ and probability at least $1 - \delta$:

$$||\boldsymbol{\Phi}(\boldsymbol{x}_{c1} - \boldsymbol{x}_{c2})|| \leq (1 + \epsilon)||\boldsymbol{x}_{c1} - \boldsymbol{x}_{c2}|| + \mathcal{O}\left(\delta\right) \ . \tag{2.40}$$

For the second and third terms in (2.39), we use Lemma 2.5.5. With probability as least $1 - \delta' = 1 - 2e^{-\frac{M}{2}}$:

$$||\boldsymbol{\Phi}(\boldsymbol{x}_1 - \boldsymbol{x}_{c1})|| \leq (2 + \sqrt{\frac{N}{M}})||\boldsymbol{x}_1 - \boldsymbol{x}_{c1}|| \leq (2 + \sqrt{\frac{N}{M}})\beta \ . \tag{2.41}$$

And with probability as least $1 - \delta'' = 1 - 2e^{-\frac{M}{2}}$:

$$||\boldsymbol{\Phi}(\boldsymbol{x}_2 - \boldsymbol{x}_{c2})|| \leq (2 + \sqrt{\frac{N}{M}})||\boldsymbol{x}_2 - \boldsymbol{x}_{c2}|| \leq (2 + \sqrt{\frac{N}{M}})\beta \ . \tag{2.42}$$

With $M \geq \mathcal{O}\left(\frac{D}{\epsilon^2} \log \frac{Tr_c}{\delta}\right)$, a small ϵ, and some reasonable values for r, T, L, we can show that $\delta' < \delta$ and $\delta'' < \delta$.

Now we can combine the the upper bound of each term in (2.39) and conclude that with

$M \geq \mathcal{O}\left(\frac{D}{\epsilon^2}\log\frac{Tr_c}{\delta}\right)$ and probability as least $1-3\delta$:

$$\begin{aligned}||\mathbf{\Phi}(\boldsymbol{x}_1-\boldsymbol{x}_2)|| &\leq (1+\epsilon)||\boldsymbol{x}_{c1}-\boldsymbol{x}_{c2}|| + (4+2\sqrt{\frac{N}{M}})\beta + \mathcal{O}(\delta) \\ &\leq (1+\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| + (1+\epsilon)||\boldsymbol{x}_{c1}-\boldsymbol{x}_1|| + (1+\epsilon)||\boldsymbol{x}_2-\boldsymbol{x}_{c2}|| \\ &\quad + (4+2\sqrt{\frac{N}{M}})\beta + \mathcal{O}(\delta) \\ &\leq (1+\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| + (6+2\sqrt{\frac{N}{M}}+2\epsilon)\beta + \mathcal{O}(\delta)\ .\end{aligned} \tag{2.43}$$

We have shown the derivation of the upper bound. The derivation of the lower bound follows similar steps. We state the following lemma for the pairwise preserving property:

Lemma 2.5.7 *Assume a Lipschitz continuous mapping $G(\cdot,\cdot): \mathbb{R}^D \times \mathbb{R}^{L-D} \to \mathbb{R}^N$ that has Lipschitz constant T and satisfies: $||G(\boldsymbol{c},\boldsymbol{v}_1)-G(\boldsymbol{c},\boldsymbol{v}_2)|| \leq \beta,\ \forall \boldsymbol{c} \in \mathcal{B}^D(r_c), \boldsymbol{v}_1,\boldsymbol{v}_2 \in \mathcal{B}^{L-D}(r_v)$, where $\mathcal{B}^L(r)$ denotes a norm ball in $\mathbb{R}^L$ with radius r: $\mathcal{B}^L(r) = \{z | z \in \mathbb{R}^L, ||z|| < r\}$ and $||\cdot||$ denotes the L_2 norm. Assume also a random matrix $\mathbf{\Phi} \in \mathbb{R}^{M\times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0,1/M)$. Given some $0<\epsilon<1$, $0<\delta<1$, if $M \geq \mathcal{O}\left(\frac{D}{\epsilon^2}\log\frac{Tr_c}{\delta}\right)$, then with probability at least $1-\delta$:*

$$||\mathbf{\Phi}(\boldsymbol{x}_1-\boldsymbol{x}_2)|| \geq (1-\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| - (6+2\sqrt{\frac{N}{M}}-2\epsilon)\beta - \mathcal{O}(\delta) \tag{2.44}$$

and

$$||\mathbf{\Phi}(\boldsymbol{x}_1-\boldsymbol{x}_2)|| \leq (1+\epsilon)||\boldsymbol{x}_1-\boldsymbol{x}_2|| + (6+2\sqrt{\frac{N}{M}}+2\epsilon)\beta + \mathcal{O}(\delta) \tag{2.45}$$

holds $\forall \boldsymbol{x}_1,\boldsymbol{x}_2 \in G(\mathcal{B}^D(r_c),\mathcal{B}^{L-D}(r_v))$.

Going from the pairwise distance preserving property to the recovery guarantee, we follow the similar steps as shown in deriving Lemma 2.5.6, which leads to the following lemma:

Lemma 2.5.8 *Assume a Lipschitz continuous mapping $G(\cdot,\cdot) : \mathbb{R}^D \times \mathbb{R}^{L-D} \to \mathbb{R}^N$ that has Lipschitz constant T and satisfies: $||G(\boldsymbol{c}, \boldsymbol{v}_1) - G(\boldsymbol{c}, \boldsymbol{v}_2)|| \leq \beta,\ \forall \boldsymbol{c} \in \mathcal{B}^D(r_c), \boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathcal{B}^{L-D}(r_v)$, where $\mathcal{B}^L(r)$ denotes a norm ball in $\mathbb{R}^L$ with radius r: $\mathcal{B}^L(r) = \{z | z \in \mathbb{R}^L, ||z|| < r\}$ and $||\cdot||$ denotes the L_2 norm. Assume also a random matrix $\boldsymbol{\Phi} \in \mathbb{R}^{M \times N}$ whose entries are sampled from an i.i.d. Gaussian $\mathcal{N}(0, 1/M)$. The compressed measurements are obtained as $\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}^* + w$, where $\boldsymbol{x}^*$ is the desired signal and w is an independent additive noise. Define $\bar{\boldsymbol{x}} = \underset{\boldsymbol{x} \in G(\mathcal{B}^D(r_c), \mathcal{B}^{L-D}(r_v))}{\arg\min} ||\boldsymbol{x}^* - \boldsymbol{x}||$ and $\hat{\boldsymbol{x}} = \underset{\boldsymbol{x} \in G(\mathcal{B}^D(r_c), \mathcal{B}^{L-D}(r_v))}{\arg\min} ||\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{x}||$.*

For some $0 < \epsilon < 1$, $0 < \delta < \frac{1}{2}$, if $M \geq \mathcal{O}\left(\frac{D}{\epsilon^2} \log \frac{Tr_c}{\delta}\right)$, then with probability at least $1 - \delta$:

$$||\boldsymbol{x}^* - \hat{\boldsymbol{x}}|| \leq \frac{5 + 2\sqrt{\frac{N}{M}} - \epsilon}{1 - \epsilon} ||\boldsymbol{x}^* - \bar{\boldsymbol{x}}|| + \frac{6 + 2\sqrt{\frac{N}{M}} - 2\epsilon}{1 - \epsilon} \beta + \frac{2}{1 - \epsilon} ||w|| + \mathcal{O}(\delta) \ . \quad (2.46)$$

Lemma 2.5.8 corresponds to the first part of our main theorem. The number of required measurements is linear with D which is usually much smaller than the full latent variable dimension L. When the number of measurements increases, we can apply Lemma 2.5.6 directly with $r = \sqrt{r_c^2 + r_v^2}$ since $\mathcal{B}^D(r_c) \times \mathcal{B}^{L-D}(r_v) \subseteq \mathcal{B}^L(\sqrt{r_c^2 + r_v^2})$, and arrive at the second part of our main theorem. We have now finished the entire proof of Theorem 2.5.1.

2.6 Chapter summary

This chapter proposes an algorithm of using generative models to solve compressive sensing inverse problems. This method is fast to carry out, by exploiting a projector network, and is stable under high compression factor, by putting constraints on the latent variable space. It consistently produces high-quality images even when the observations are highly compressed. The proposed algorithm can be easily generalized to solve inverse imaging problems besides CS recovery.

CHAPTER 3
MACHINE LEARNING IN THE COMPRESSED DOMAIN: GESTURE RECOGNITION

In this chapter, we demonstrate how machine learning can be used in combination of compressing technologies to perform energy-efficient signal processing tasks. Specifically, we propose a novel appearance-based gesture recognition algorithm using compressed domain signal processing techniques. Gesture features are extracted directly from the compressed measurements, which are the block averages and the coded linear combinations of the image sensor's pixel values. We also improve both the computational efficiency and the memory requirement of the previous DTW-based K-NN gesture classifiers. Both simulation testing and hardware implementation strongly support the proposed algorithm.

3.1 Introduction and related work

Hand gesture recognition is continuously evolving in how systems on chip (SoCs) interact with users. To achieve power efficiency, these SoCs turn into idle mode when not being used and "wake up" when users are detected. The detection of an user's present requires the sensor front-ends to be perpetually ON, thus making low power consumption an important design criteria. As cameras have become default devices embedded in many systems, a camera-based hand gesture recognition system is suitable for providing stimulus for system wake up.

Based on image outputs from a camera, most existing algorithms work directly in the pixel domain [57, 58]. The majority of the work can be divided into three stages. First, the hand region is extracted from the image using techniques such as background extraction, skin color detection, and contour detection. Second, the motion of the gesture is characterized by features. The common types of features include difference image, motion centroid, optical

flow, and motion vectors. At the last stage, these features are sent to a classifier. Dynamic time warping (DTW) with K nearest neighbors (K-NN), hidden Markov models, and neural networks have all been implemented and showed promising result.

Aforementioned algorithms require a significant amount of energy in the analog to digital (A/D) conversion of each pixel of the image sensor. Reducing the number of sensing measurements plays an important role of energy saving. Recent development in compressive sensing and target recognition in the compressed domain [17, 59, 60] improved the performance and energy efficiency of the overall process of data acquisition, feature extraction and recognition. These works suggest us taking coded combinations of the pixel vales and characterizing the gesture motion directly from a few compressed measurements.

In this chapter, we propose an appearance-based gesture recognition algorithm for system wake up. The gesture motion is captured by a sequence of difference images. Each difference image passes through two layers of compression to reduce its resolution and to be transferred to the compressed domain. The parameters of the motion are then directly extracted from the compressed domain and used as features for classification. To the authors' knowledge, our work is the first in gesture recognition using compressive sensing techniques. We also enhance the previous DTW-based K-NN classifiers [61, 62], allowing them to cooperate with clustering and dimension reduction techniques, and therefore, improve both the computational and the memory efficiency of time series classification. In a hardware co-design, we implement the compression in the sensor front-end and motion parameter estimation in the mixed signal domain. The testing results show significant energy saving over previous works [63, 64].

3.2 Algorithms

The block diagram of our system is shown in Figure 3.1. Difference images are capable of capturing gestures containing significant motions. We pass each difference image through two layers of compression. In the first layer, the resolution is reduced by dividing the whole

image into several blocks and taking the average of each block. In the second layer, we take coded combinations of these block-averaged pixels. We estimate the center of the motion directly from these compressed measurements. These motion centers are passed to a classifier for gesture recognition.

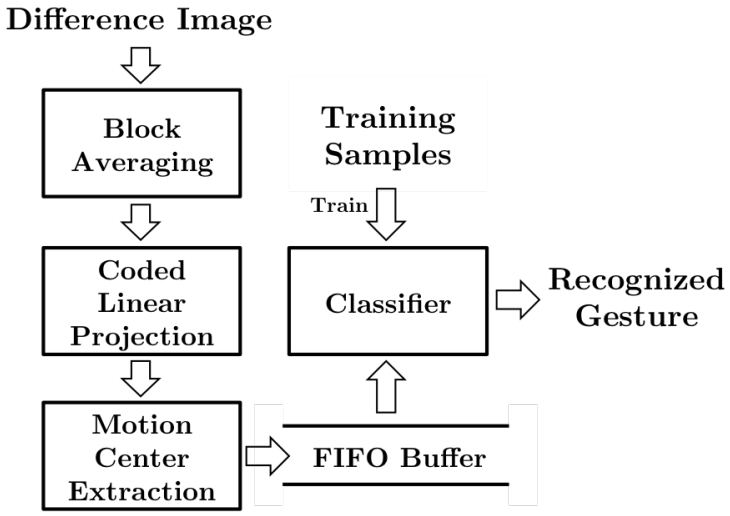


Figure 3.1: Block diagram of the proposed algorithm

3.2.1 Two layers of compression

Denote F_i as the ith full resolution image output from the camera of size $W \times H$. The difference image D_i (Figure 3.2.a) of two consecutive frames is calculated as $D_i = |F_{i+1} - F_i|$.

In the first compression layer, the difference image is divided evenly into blocks of size B by B. The average of the pixel values in each block is taken, resulting in a block-compressed difference image of size W/B by H/B (Figure 3.2.b). We vectorize this low-resolution difference image and denote it as $y_i \in \mathbb{R}^N$.

In the second layer of compression, we chose a random matrix Φ of size M by N as the coded measuring matrix. Each entry of Φ is uniformly chosen from $\{+1, -1\}$. The projection of the vectorized low-resolution difference image in the compressed domain is

calculated as:

$$\hat{y}_i = \Phi y_i = \Phi \Psi Y_i \tag{3.1}$$

Each entry in $\hat{y}_i \in \mathbb{R}^M$ is a random linear combination of all the entries in y_i. Y_i is the vectorized original difference image D_i. Ψ is the block averaging matrix of size N by $W \times H$.

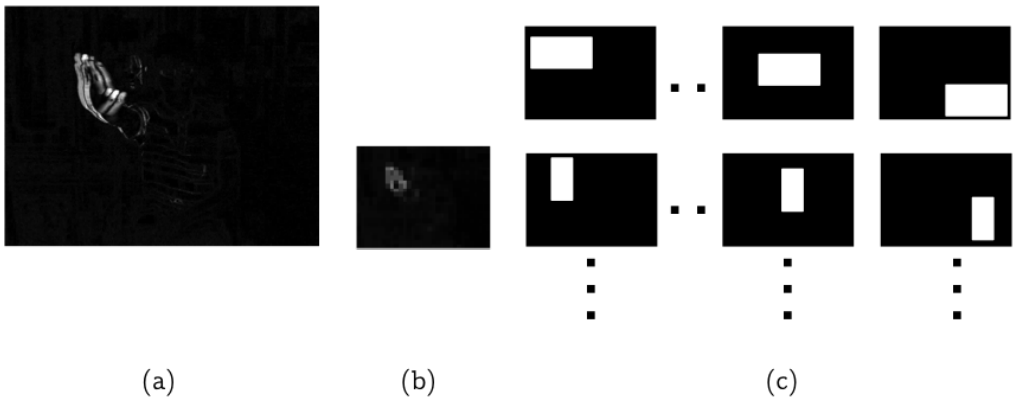


Figure 3.2: (a) Full-resolution difference image D_i; (b) Block averaged difference image; (c) Matching templates in the uncompressed domain. Rectangle sizes differ among rows and centers of the rectangles differ among columns.

3.2.2 Motion center extraction in the compressed domain

In the uncompressed low-resolution domain, the hand region in the difference image can be captured by a template shown in Figure 3.2.c. The template (of size W/B by H/B) has uniform non-zero values within the small rectangular region and zeros elsewhere. To locate the hand region, we construct a set of vectorized templates $X(\alpha, r)$, where $\alpha \in \mathbb{R}^2$ represents the x-y coordinates of the center of the small rectangle, and r represents different rectangle sizes. The variation in sizes is to adapt to the change of the hand size seen by the camera when users are at different locations. The center of the hand motion is extracted by solving

$$(\alpha^*, r^*) = \arg\min_{\alpha, r} ||y_i - X(\alpha, r)||_2 \tag{3.2}$$

The collection of templates forms a manifold in $\mathbb{R}^N$ with intrinsic parameters α and r. Using the result from [17] and [11], we can directly extract the motion centers in the compressed domain. That is, for

$$(\hat{\alpha}^*, \hat{r}^*) = \arg\min_{\alpha,r} ||\hat{y}_i - \Phi X(\alpha, r)||_2 \tag{3.3}$$

$(\hat{\alpha}^*, \hat{r}^*) \approx (\alpha^*, r^*)$ with high probability for some $M \ll N$. The block averaging layer reduces the possible choices of r, and techniques such as matched filtering can be applied to efficiently solve (3.3).

3.2.3 Gesture classifier training

Once the motion center $\hat{\alpha}^*$ is extracted from each compressed frame, a motion gesture is represented by a sequence of $\hat{\alpha}_i^*$. At this point, the gesture recognition problem has become a time series classification problem. Two important factors need to be considered for motion gesture recognition. First, two samples of the same gesture class can have different length. For instance, let the gesture class be writing a letter "Z" in front of the camera. Depending on how fast an user completes the gesture, the collected time series can contain different number of points. Second, the gesture classifier should remain invariant to reasonable time shifts and local distortions in the gesture samples. In the "Z" gesture example which contains three straight strokes, an user can write each stroke with a different speed. The slower stroke generates more data points while the faster stroke creates fewer. In order to successfully classify all these samples as the same gesture, reasonable time shift and local distortions among different time series need to be considered as invariant by the classification algorithm.

To design similarity measures for gesture time series of different lengths and variations, we apply the dynamic time warping (DTW) algorithm. The DTW algorithm calculates the pointwise optimal match between two time series under the following restrictions [65]:

- Every point from the first time series $\mathbf{T}^1$ must be matched with one or more points from the other sequence $\mathbf{T}^2$, and vice versa.

- The first point in the first time series must be matched with the first point in the other time series: $(1, 1) \in \mathcal{M}$.
- The last point in the first time series must be matched with the last point in the other time series: $(L_1, L_2) \in \mathcal{M}$.
- The matched pair of points from the two sequence must be monotonically increasing. That is, for every two pairs of matched points $(i_1, i_2) \in \mathcal{M}$ and $(j_1, j_2) \in \mathcal{M}$, if $j_1 > i_1$, then $j_2 \geq i_2$. And if $j_2 > i_2$, then $j_1 \geq i_1$.
- The time indices of every pair of matched points must satisfy locality constraint. That is, $\mid i_1 - i_2 \mid \leq w, \ \forall (i_1, i_2) \in \mathcal{M}$, where w is the length of the locality window size and should satisfy $w \geq \mid L1 - L2 \mid$.

When one point $\mathbf{T}^1(i_1)$ in the first time series is matched with $\mathbf{T}^2(i_2)$ in the second time series, the distance between these two points are calculated. Denote this pointwise distance as $d_p(\mathbf{T}^1(i_1), \mathbf{T}^2(i_2))$. The distance between the two time series according to a matching set is defined as the sum of all the pointwise distance between the matched pairs:

$$d_{\mathcal{M}} = \sum_{i_1, i_2 \in \mathcal{M}} d_p(\mathbf{T}^1(i_1), \mathbf{T}^2(i_2)). \tag{3.4}$$

The DTW distance between two time series is then defined as the minimum $d_{\mathcal{M}}$ over all valid matching sets:

$$d_{DTW} = \min_{\mathcal{M}} d_{\mathcal{M}} = d_{\mathcal{M}_{DTW}}. \tag{3.5}$$

This minimization problem can be solved efficiently using dynamic programming. Both the DTW distance d_{DTW} and its corresponding pointwise matching set $\mathcal{M}_{DTW}$ can be calculated in $\mathcal{O}(L^2)$ time with $\mathcal{O}(L^2)$ memory requirement, where L is the length of the time series.

As DTW defines a distance measure between two time series, we can directly plug it into existing classification algorithms. It has been shown that DTW-based classifiers

perform well for dataset containing limited amount of samples [66]. Traditional DTW-based classifiers use DTW [67] as the distance measure between two sequences of different lengths, and use K-NN method for classification. The memory and computational requirements of this classification algorithm thus grow linearly with the size of training set.

To reduce the number of DTW calculations in the recognition stage, we perform K-means clustering in the training dataset to form "super samples". The distance between an individual sample and a super sample is measured using DTW. In each iteration, the super samples are updated as the average of all the samples within their clusters. DTW barycenter averaging (DBA) [68] is used as the averaging method. Given a data set of n time series $\mathbf{T}^1, \mathbf{T}^2, \ldots, \mathbf{T}^n$ to be averaged, we first initialize a new time series $\mathbf{S}$ of size τ. τ can be chosen as either the median or the mean length of all the sequences to be averaged. Then we run DTW between $\mathbf{S}$ and every $\mathbf{T}^i$ in the dataset, and store the point matching information $\mathcal{M}_i$. After collecting all the matching information, we go through every point in $\mathbf{S}$. For each point $\mathbf{S}(j)$ in the time series, we find all the points it has been matched to and assign its value as the mean value of all these matched points. The DBA algorithm is summarized in Algorithm 3.1. The result of the DBA algorithm can be sensitive to the initialization of sequence $\mathbf{S}$. In practice, we can cascade this algorithm multiple times, using the result from the previous run as the initialization sequence. For our application, we observe that running DBA as few as 5 times is enough to return stable and satisfactory results.

Algorithm 3.1 DTW Barycenter Averaging (DBA)

Require: The initial average sequence $\mathbf{S}$ of size τ.
Require: n sequences $\mathbf{T}^1, \mathbf{T}^2, \ldots, \mathbf{T}^n$ to be averaged.
  for $i = 1$ to n **do**
    Run $DTW\ (\mathbf{S}, \mathbf{T}^i)$.
    $\mathcal{M}_i \leftarrow$ pairwise matching information between $\mathbf{S}$ and $\mathbf{T}^i$.
  end for
  for $j = 1$ to τ **do**
    $\mathcal{P}_j \leftarrow$ All points matched to $\mathbf{S}(j)$ based on $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_N$.
    $\mathbf{S}(j) =$ mean($\mathcal{P}_j$).
  end for

One of the major difficulties in time series classification comes from the different lengths of the samples. We notice that DBA can find clustering centers of an arbitrary length set by the user. The pairwise matching information in DTW also provides a way to rescale the length of time sequences. Therefore, we propose a DTW length rescale algorithm, shown in Algorithm 3.2. Given a set of previously calculated super samples $\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^k$ all of which are of length τ, and a time series $\mathbf{T}$ to be rescaled, we first find one super sample $\mathbf{S}^*$ that is the closest to $\mathbf{T}$ under the DTW distance measurement. We also store the pointwise matching information $\mathcal{M}$ between $\mathbf{T}$ and $\mathbf{S}^*$. We then initialize a new time series $\mathbf{T}'$ of size τ and start going through each point $\mathbf{S}^*(j)$ in $\mathbf{S}^*$. If $\mathbf{S}^*(j)$ is only matched to one point in $\mathbf{T}$, then this matched point is copied to $\mathbf{T}'(j)$. If $\mathbf{S}^*(j)$ is matched to multiple points in $\mathbf{T}$, then among these matches we choose the one closest to $\mathbf{S}^*(j)$ and copy its value to $\mathbf{T}'(j)$. In this way, we perform local interpolation and contraction on $\mathbf{T}$ based on the matching information $\mathcal{M}$, and produce a new time series $\mathbf{T}'$ which is the rescaled version of $\mathbf{T}$ with length τ.

Algorithm 3.2 DTW Length Rescaling (DLR)

Require: k "super samples" $\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^k$ of length τ, calculated using K-means with DTW and DBA.
Require: Sequence $\mathbf{T}$ to be rescaled to length τ.
  $\mathbf{S}^* = \underset{\mathbf{S} \in \mathbf{S}^1, \ldots, \mathbf{S}^k}{\arg\min}\ DTW(\mathbf{S}, \mathbf{T})$.
  $\mathcal{M} \leftarrow$ pairwise matching information between $\mathbf{S}^*$ and $\mathbf{T}$.
  Initialize $\mathbf{T}'$ of length τ.
  for $j = 1$ to τ **do**
    if $\mathbf{S}^*(j)$ is matched to multiple points $\mathbf{T}(i), \mathbf{T}(i+1), \ldots, \mathbf{T}(i+l)$, according to $\mathcal{M}$, **then**
      $\mathbf{T}'(j) = \underset{\mathbf{T}(m) \in \mathbf{T}(i), \ldots, \mathbf{T}(i+l)}{\arg\min} d_p(\mathbf{S}^*(j), \mathbf{T}(m))$.
    else
      $\mathbf{T}'(j) = \mathbf{T}(i)$, where $\mathbf{T}(i)$ is the only matching point to S_i^*.
    end if
  end for

Using Algorithm 3.2, we rescale all the training sequences to the same length τ. Since each motion center in the time sequence contains both x and y coordinates, each gesture

sample is now of size 2 by τ. We vectorize the samples by cascading all the y coordinates after the x coordinates, transferring the time series classification problem to a traditional classification problem in $\mathbb{R}^{2\tau}$. Various dimension reduction techniques and multi-class classification algorithms can then be implemented. The block diagram of the complete training procedure is shown in Figure 3.3.

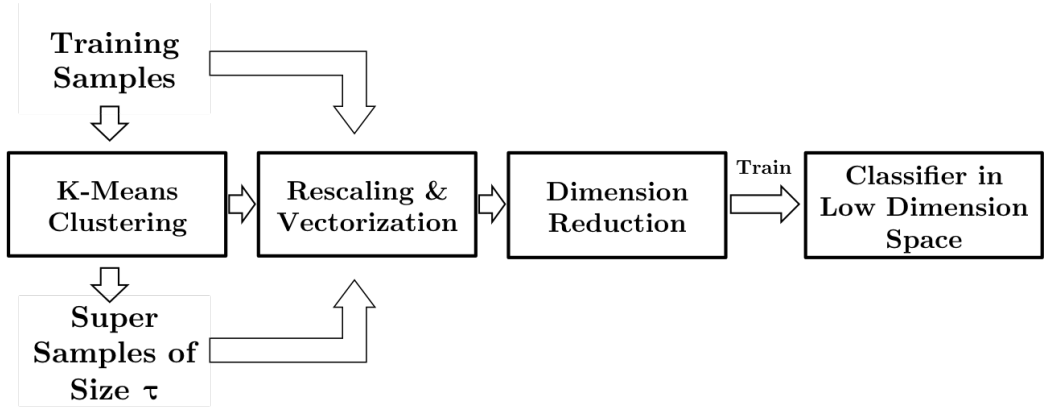


Figure 3.3: Block diagram of training the gesture calssifier

3.2.4 Gesture recognition

In a real-time system, the extracted motion centers are stored in a FIFO buffer of length L. To recognize the gesture, we first rescale the buffer data sequence based on the learned super samples using Algorithm 3.2. Within this step, open-ended DTW is used for pairwise matching in order to automatically separate the gesture-like data from the noise at both ends of the buffer. The new gesture-like sequence of length τ then goes through vectorization and dimension reduction before being sent to the trained classifier. The block diagram of the complete recognition procedure is shown in Figure 3.4.

Compared to the traditional DTW-based K-NN classifiers, our classifier significantly reduces the number of DTW calculations in the recognition stage and is still able to exploit the structure of the entire training set. In our experiments, most of the gestures can be well separated in a very low dimension, making the dimension reduction and the low-dimensional classifier computationally efficient.

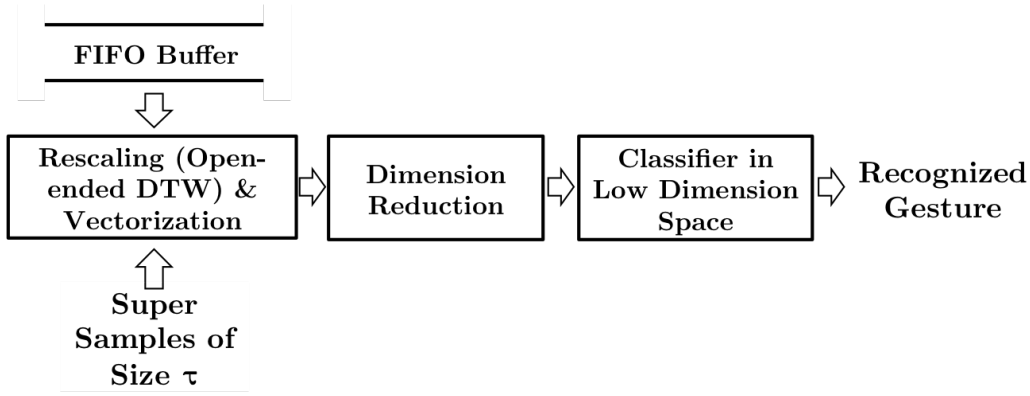


Figure 3.4: Block diagram of gesture recognition

3.3 Testing and results

3.3.1 Number of compressed measurements

To gain better insight for the choice of the number of compressed measurement M, we explore its relationship with the accuracy of motion center extraction. We run the extraction algorithm with one video of gesture "Z" (Figure 3.5.a). In the block averaging layer, difference images of size 480 by 640 are compressed by blocks of size 16 by 16. We extract the motion centers from these block-averaged difference images by solving (3.2). Shown in Figure 3.5.b, the three segments of the gesture are clearly distinguished in the path of the motion centers. This result is used as the ground truth for comparing M.

We then use only 250 compressed measurements ($M = 250$) and the motion centers are extracted by solving (3.3), and are plotted in Figure 3.5.c. The apparent similarity between this plot and 3.5.b verifies the theory. For each value of M, we calculate the average motion center error per frame in the compressed domain. The "L" shape of the curve indicates that $M = 250$ is the threshold for nearly error-free motion parameter estimation, granting us another factor of 5 compression rate. This "threshold" behavior is consistent with the classic results from compressed sensing presented in [17, 59, 11]. The accurate motion center extraction in the compressed domain provides the foundation of high recognition rate.

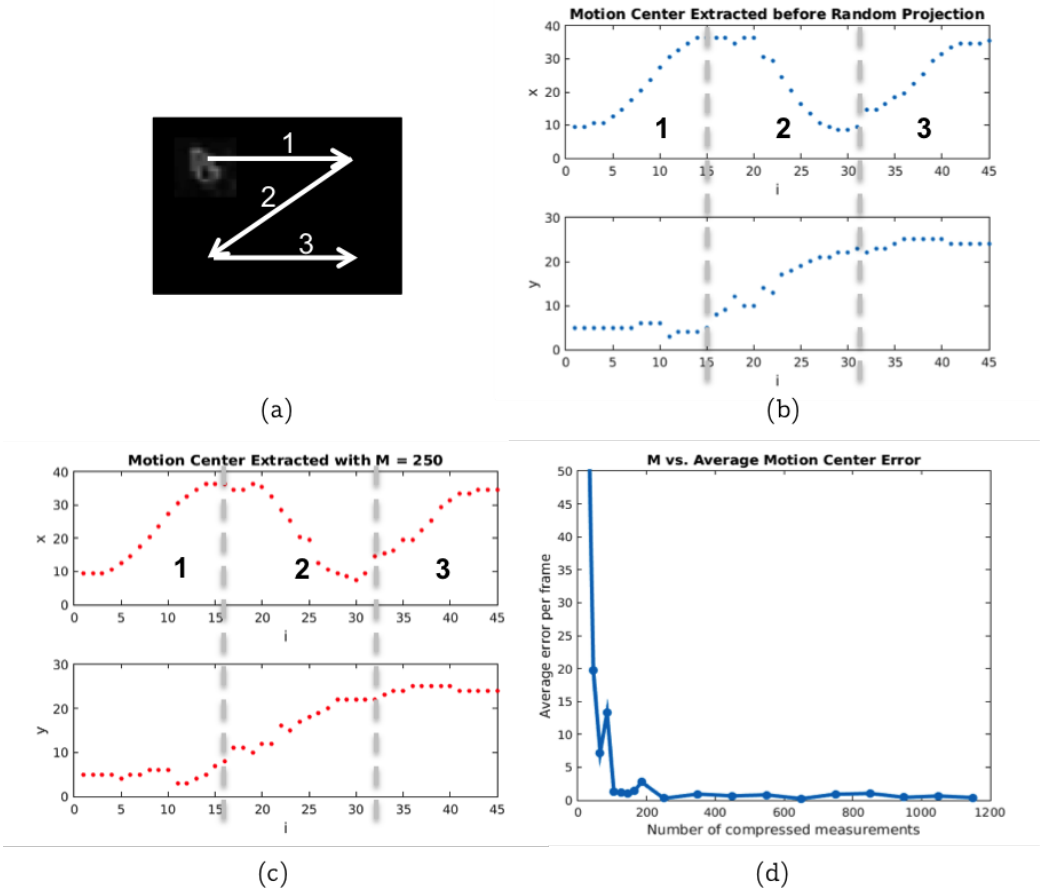


Figure 3.5: MATLAB simulation results of estimating motion centers using different numbers of compressed measurements M. (a) Hand motion of gesture "Z" diveded into three segments; (b) Motion center extracted before random projection by solving equation (3.2); (c) Motion center extracted from 250 compressed measurements by solving (3.3); (d) Average error of motion center extracted in the compressed domain when compared with (b).

3.3.2 SKIG dataset

We test the overall algorithm on the public-available SKIG dataset [69]. We select 5 classes containing significant motion in the x-y plane: Circle, Triangle, Wave, Z, and Cross. In each class, we select 70 well-illuminated samples and randomly divided them into training (55 samples) and testing (15 samples) sets. We crop the training videos so that the gesture motion fills the entire video. Since each frame is of size 240 by 320, in the block compression layer, we use blocks of size 10 by 10, and we set $M = 200$ in the random projection layer.

During the training stage, we calculate one super sample in each gesture class. As the

lengths of the gesture videos vary from 48 to 236 frame, we choose τ to be the average length 116. After rescaling and vectorizing all the training sample, we apply PCA and use the first 3 principle components. The gesture samples are well separated in $\mathbb{R}^3$, as shown in Figure 3.6.a. For simplicity, we model each gesture class' distribution as a multivariate Gaussian. A gesture is assigned to the class with the highest likelihood beyond a rejecting threshold. The resulting classifier has decision boundaries of ellipsoid shapes.

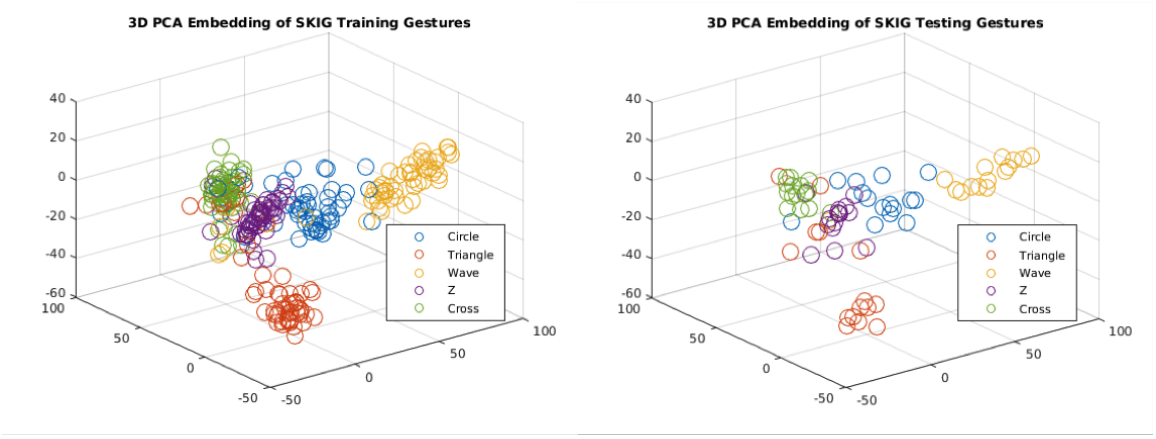


Figure 3.6: (a) PCA embedding of SKIG training samples in $\mathbb{R}^3$; (b) PCA embedding of SKIG testing samples in $\mathbb{R}^3$.

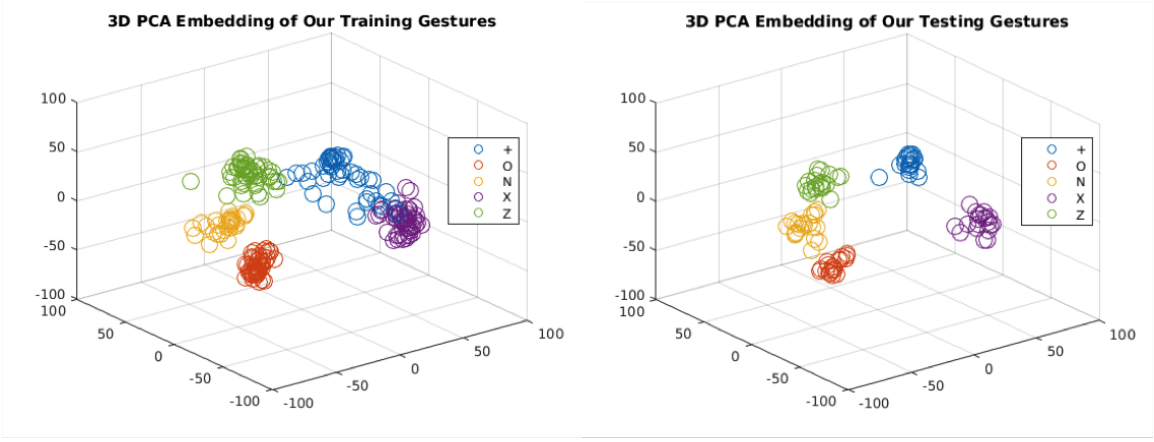
Following the proposed recognition procedure, we show the testing samples' PCA embedding in Figure 3.6.b. The recognition rate is shown in Table 3.1. The relatively low recognition rate for triangle gestures is caused by the longer sample videos that usually contain more than 200 frames. Rescaling them results in significant downsampling, and some of these downsampled data overlap with the "Z" gesture samples in the $\mathbb{R}^3$ embedding. Our classifier has comparable performance with a 5-NN classifier for all other classes.

Gesture Type	Circle	Triangle	Wave	Z	Cross
Recognition Rate (Our Classifier)	93.3%	73.3%	93.3%	80%	93.3%
Recognition Rate (DTW 5-NN)	93.3%	93.3%	93.3%	100%	100%

Table 3.1: Recognition rates of SKIG gesture dataset

3.3.3 Real-time OpenCV simulation

We simulate a real-time system using OpenCV and a webcam as the image sensor. [1] Each frame is of size 480 by 640. In the block compression layer, we use blocks of size 20 by 20 and set $M = 200$ in the random projection layer. We specify 5 different gesture classes: "+", "O", "N", "X", and "Z", each containing 50 samples. Following the similar training procedure as performed on the SKIG dataset, we transfer each gesture sample into a point in $\mathbb{R}^3$, plotted in Figure 3.7.a, and train a Gaussian maximum likelihood classifier.



(a) (b)

Figure 3.7: (a) PCA embedding of our training samples in $\mathbb{R}^3$; (b) PCA embedding of our testing samples in $\mathbb{R}^3$.

To calculate the recognition rate, we perform 20 gestures per class in front of the webcam. These testing samples' PCA embedding in $\mathbb{R}^3$ is shown in Figure 3.7.b. We calculate the false detection rate by performing 50 unspecified gestures. Table 3.2 shows both the recognition rate and the false detection rate. The simulation result strongly verifies our algorithm.

Gesture Type	+	O	N	X	Z
Recognition Rate	100%	100%	100%	95%	100%
False Detection Rate	4%	0%	4%	2%	0%

Table 3.2: Real-time OpenCV simulation results

[1] Our OpenCV (C++) demo code is available at www.kylexu.net/cs-gesture-recog

3.3.4 Hardware implementation

We implement the proposed algorithm in a gesture recognition system powered by solar energy, shown in Figure 3.8. With 400 compressed measurement, this system achieves greater than 80% accuracy and consumes only $95mJ$ of energy per frame [5]. In this system, the random projection is simulated in the MCU and the classifier used DTW-based 1-NN method. We further combine both compression layers into the camera front-end, and implement motion center extraction in the mixed signal domain. Testing results show the energy consumption reduced to $1.3\mu J$ per frame.

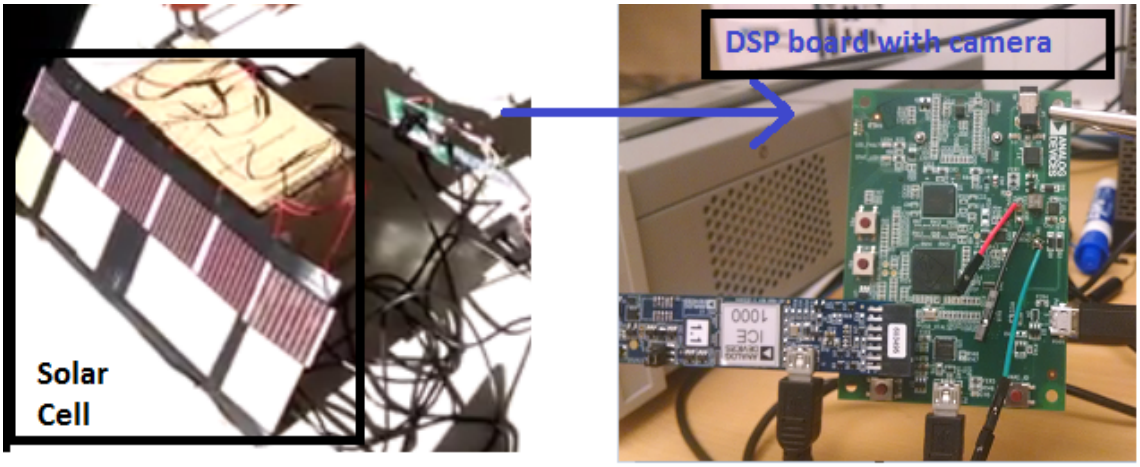


Figure 3.8: Our light-powered smart camera system.

3.4 Chapter summary

We proposed an energy-efficient appearance-based gesture recognition algorithm in the compressed domain. The major saving of power comes from the two layers of compression that reduce the resolution of the image sensor by a factor of more than 1000. Our proposed gesture classifier significantly reduces the number of DTW calculations and the memory requirements of the traditional DTW-based K-NN classifiers while preserving the structure of the full training dataset. Our algorithm has been verified by both simulation testing and hardware co-design.

CHAPTER 4
MACHINE LEARNING IN HARDWARE CONTROL: AI-ASSISTED POWER AMPLIFIER

In this chapter, we demonstrate how machine learning can be incorporated into the design of hardware control algorithms. Advanced power amplifier (PA) architecture is critical for 5G communications which require the PA to have sufficient output power, high linearity, and high energy efficiency. The time-variant operational environment further demands the self-reconfigurability in the PA design. Recent developments in bandit-problems and reinforcement learning (RL) have led to data-driven control algorithms. This chapter presents various RL-based algorithms for Doherty PA control, which achieve robust adaptive operation over environmental changes. Multiple RL frameworks are incorporated in the control algorithms including multi-armed bandit (MAB), continuum-armed bandit (CAB), contextual-bandit (CB), and actor-critic with experience replay (AC). The control algorithms based on the latter three frameworks leverage on the prior information about the Doherty PA's characteristics to improve learning efficiency. In our simulation test where the optimal policy needs to be adjusted due to transmitter impedance matching, the MAB-based control learns the optimal policy within 25,000 samples. The control algorithm based on CAB and CB learns the optimal policy within 5,000 samples. The fastest learning rate is achieved by the control algorithm based on AC, which learns the optimal policy within 1,500 samples.

4.1 Introduction

The rapid growth of fifth-generation (5G) communications has posed increasingly demanding performance challenges on future wireless front-ends, including wide and non-contiguous bandwidth, sufficient transmitter output power, adequate receiver, dynamic range, high linearity, and high energy efficiency. [70, 71, 72]. Advanced transmitter (TX)

and power amplifier (PA) architectures are highly desired for communication applications where high peak-to-average power ratio waveforms are extensively employed. Recently, the Doherty TX/PA architecture has gained much attention due to its high efficiency at power back-off (PBO), large modulation bandwidth, and low baseband overhead [73, 74, 75, 76]. However, the PBO efficiency and the linearity of Doherty TXs/PAs are highly sensitive to the auxiliary (Aux) PA onset power and the load modulation between the Main and Aux paths. The Doherty load modulation relationship often varies drastically over antenna's impedance mismatch, necessitating complicated calibrations over single-branch PAs [73, 74, 75]. It is reported that re-configuring the Main/Aux PA settings in a Doherty PA can restore the PA performance under different antenna voltage standing wave ratios (VSWRs) [75].

Circuit and system level self-testing, calibrations, and reconfigurations to enable front-end adaptivity and performance restoration also becomes essential for complex mobile applications. Wireless front-ends often need to operate in congested and constantly varying electromagnetic and thermal environments while still maintaining high performance and reliable wireless links. Although extensive work has been done on built-in-self-testing and online front-end calibrations, existing approaches often rely on extensive back-end computations and exhaustive sweeps on possible circuit configurations to control the front-end settings and then optimize front-end performance [77]. These approaches are not suitable for many emerging applications which require ultra-low latency and fast response capabilities. This is particularly an issue for PAs in 5G multiple-input and multiple-output (MIMO) systems. The PA's performance, including power gain, linearity, efficiency, and reliability, highly depends on its load impedance [75, 78]. MIMO antenna array operations inevitably introduce cross-element coupling and result in different load mismatch (VSWR) for each PA. Meanwhile, many 5G applications require ultra-low latency, leaving exceedingly limited latency budget for PA load/performance calibration.

In this chapter, we propose control algorithms for a mm-wave Doherty PA based on artificial intelligence (AI). We aim to achieve the maximum linearity in the PA's gain

response while maintaining high efficiency. The recent developments in AI have inspired new approaches for designing adaptive systems. Data about the environment and the system's interval behavior is collected during the system's operation. The AI algorithm characterize the system based on the collected data and "learns" to control the system. Within the AI frameworks, reinforcement learning (RL) which learns how to act optimally in various observable environmental states can be naturally extended to PA control algorithms. Our control algorithms are based on prominent bandit and RL frameworks such as multi-armed bandit, continuum-armed bandit, contextual bandit, and actor-critic with experience replay. These RL-based approaches provide the PA self-reconfigurability and lead to robust adaptive operation over environmental changes. The control algorithms based on the latter three frameworks also leverage on the prior information about the Doherty PA's characteristics to improve learning efficiency.

The rest of the chapter is organized as follows. In Section 4.2, we discuss related work in both PA control and general RL-based control algorithms. Section 4.3 provides an overview of the architecture of the AI-assisted mm-wave Doherty PA. Our proposed control algorithms are presented in Section 4.4. Three control algorithms are discussed, each followed by testing results and comparisons. In our simulation test, the optimal policy needs to be adjusted due to VSWR changes, and our control algorithm can learn the optimal policy within 1,500 samples. Section 4.5 concludes the chapter.

4.2 Related Work

4.2.1 Circuit-Level PA control

The previous work on improving the PA's performance can be summarized into two categories. The first category is to design a reconfigurable PA at the circuit level and provide adjustable hardware settings to control the PA's behavior. Various sensors are incorporated into the architecture to measure power, efficiency, and temperature either explicitly or implicitly. Based on the measurements, the control unit adjusts the PA settings to match

certain performance criteria. One common goal of this approach is to correct the specification mismatch inevitably introduced in the manufacture of the integrated circuits. The PA architecture design which aims to calibrate the PA's behavior is known as self-healing [79, 80, 81, 82, 83, 84]. Although the self-healing architecture contains a feed-back control loop, the performance optimization is carried out not during the PA's in-field operation, but rather in a designated calibration stage. A testing signal is sent to the PA during the calibration process. The control algorithm optimizes the PA's performance by either sweeping through the PA settings [80, 81, 83, 84] or applying gradient-based optimization algorithms [79, 82].

In order to achieve true in-field PA performance optimization, several changes need to be introduced to the self-healing architecture. Self-healing often aims to optimize multiple performance aspects simultaneously, which may not be achievable given the time and computational budget of an online control algorithm. Sensor design is also crucial for in-field PA control, as the sensors must be able to provide timely and accurate measurements. Furthermore, the efficiency of the control algorithm needs to be improved in order to adapt the control policy to environmental changes. As demonstrated later in this chapter, the efficiency improvement often comes from leveraging the prior knowledge of the PA's behavior.

Dynamic PA hardware control to improve power efficiency has been explored previously in [85, 86, 87], where both the PA architectures and the control settings are relatively simple, and the algorithm can be realized at the circuit level. In-field linearization and efficiency optimization for Doherty PA was proposed in [78, 88], where the PA's behavior is approximated by polynomial functions within the feedback control unit.

Other dynamic linearization techniques for Doherty PAs have focused on digital predistortion (DPD). DPD is another popular class of methods to enhancee PAs' performance [89, 90, 91]. Input signals are modified before passing through the PA in order to compensate the PA's distortion, thus achieving extended linear response and high power efficiency. Mathematically, DPD models the PA's distortion as a complex function and seeks to construct

a system that approximates its inverse function. For a memoryless PA whose output only depends on the current input and not the history of the inputs, the PA's behavior can be fully described by gain response curves and phase response curves [92]. The DPD designs for the memoryless PA model were presented in [93, 94, 89], which is suitable for narrow bandwidth systems [92].

One advantage of the DPD is being able to address the memory effect of the PA system, where the PA's output is assumed to be determined by the input history. The memory effect causes the PA's transfer function to vary with the modulation frequency [95]. The Wiener model [96] and memory polynomial model [97] are two popular memory effect models, whose corresponding DPD designs are presented in [98, 99, 100]. Recent developments in neural networks also lead the researchers to leverage on the neural networks' powerful function approximation ability to model both PAs and their corresponding DPD systems [101, 102, 103, 104, 105].

DPD techniques and our proposed work complement each other. Our control algorithm optimizes the PA's performance by updating the hardware settings, while DPD modifies the input signal. Our control algorithm performs independently from DPD, while DPD treats the PA with our control algorithm running as an unity. The proposed control minimizes the PA distortion in the hardware level, reducing the workload of the DPD unit. On the other hand, DPD fixes the memory effect of the PA and compensates for the memoryless assumption in our proposed algorithms. Both the proposed PA control and the online DPD algorithms [106, 107, 108] are fully adaptive to environmental changes.

Recent developments in RL provide new insights for the control algorithms [109, 110, 111]. Within the field of RL, bandit problems focus on choosing the actions that generate maximum expected reward. The bandit setup assumes the actions to have no lasting impact on the system and is, therefore, suitable for designing control algorithms for moemoryless systems. Early successes have been reported in applications such as solar panel control [112], wireless communication management [113], internet of things [114], and cyber security

[115]. When memory effect is presented in the system, promising results have been reported for designing control algorithms using Markov-decision-process(MDP)-based reinforcement learning frameworks [116, 117, 118].

To the authors' knowledge, our work is the first fully adaptive hardware control algorithm for Doherty PA linearization. It is also the first Doherty PA system with bandit/RL-based controls. The learning nature of the bandit-RL frameworks allows our system to adapt to environmental changes and maintain robust performance. We incorporate the properties of the Doherty PA into the algorithm design achieve high learning efficiency and fast adaption rate.

4.3 AI-Assisted MM-Wave Doherty PA Architecture

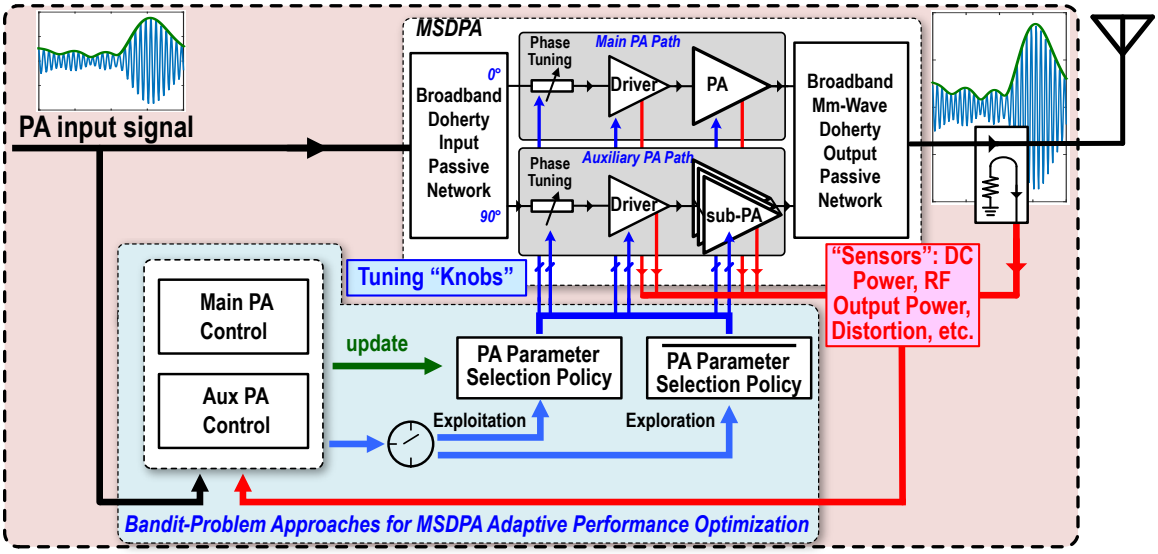


Figure 4.1: Conceptual diagram of the AI-assisted mm-wave Doherty PA. The MSDPA contains a Main and an Aux PA paths. Both inputs and the sensed outputs are sent to the control unit which runs bandit/RL-based algorithms to achieve extended linear gain region while maintaining high PA efficiency.

A conceptual diagram of the AI-assisted mm-wave Doherty PA is shown in Figure 4.1. The signal path is divided into a main PA (Main PA) path and an auxiliary PA (Aux PA) path, each providing a number of digital settings to control its behavior. The control unit adjusts these settings to achieve linear gain response and high power efficiency. As the PA's behavior

changes with environmental variables, we implement a closed-loop control to dynamically update the setting selection policy, so that the optimal performance is maintained. Detailed description about each component in the proposed system is provided below.

4.3.1 Mixed-Signal Doherty PA

The mixed-signal Doherty PA (MSDPA) [74] employs a hybrid use of an analog PA and a binary-weighted digitally controlled PA in its main and auxiliary paths, respectively. MSDPA combines the advantages of the high linearity of the main analog PA and the flexible reconfigurability of the auxiliary digital PA, while overcoming their intrinsic limitations. The MSDPA is driven by a generic complex-modulated signal that has both phase and envelope modulations. Based on the real-time AM envelope, MSDPA's Doherty operation turns on different weightings of the auxiliary digital PA. For small input envelopes, the MSDPA operates in its analog regime turning on only the main analog PA. For large input envelopes, the MSDPA operates in its mixed-signal regime. The sub-PAs in the auxiliary path are dynamically turned-on to prevent main PA from clipping or saturation. In this way the PA's operation range is extended. The overall MSDPA achieves a high linear output power and efficiency boost through its Doherty operation.

4.3.2 Main/Aux PA settings

The Doherty PA provides two sets of controls: Main and Aux path PA settings. Each control setting leads to unique gain/phase responses and PA efficiency (PAE). Figure 4.2 demonstrates the gain responses under different Main/Aux PA settings. There are 7 Main PA settings and 8 (3-bit) Aux PA settings in total. Each Main PA setting corresponds to one colored group in Figure 4.2. Given one Main PA setting, the gain responses under different Aux PA settings are plotted in the same color with different intensity. Comparing Figure 4.2.a and 4.2.b shows how the change in VSWR affects the PA's gain response. This sensitivity to the environment necessitates an efficient adaptive control algorithm.

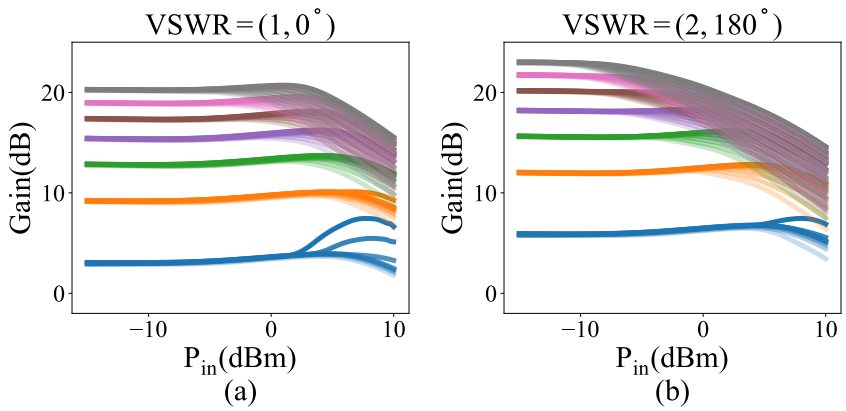


Figure 4.2: Gain responses of the proposed Doherty PA under different control setting. Main PA settings are indicated by different colors, while Aux PA settings are indicated by different intensity. (a): under standard 50ohm VSWR (b): under VSWR=$(2, \angle 180°)$

As to be discussed in Section 4.4, the PA settings are adjusted by the AI-assisted feedback control unit which aims to achieve the extended linear gain region. The control is performed in two stages, selecting the Main and Aux PA setting respectively. Under a fixed Main PA setting, the gain responses under different Aux PA settings are demonstrated in Figure 4.3.a. When the input power is low, the gain curves are flat and nearly stay the same under various Aux PA setting. This low input power region is known as the power back-off (PBO) region. As the input power increases, the gain curves start to drop. The curves under small Aux PA settings drop faster than the ones under large Aux PA settings. The phase responses and the PAE are shown in Figure 4.3.b and Figure 4.3.c respectively. Although the longest linear gain region is achieved by fixing the Aux PA setting as 7, the PAE of this strategy is significantly degraded. The shape of the gain curves and the PAE curves suggests that the Aux PA setting should be dynamically adjusted based on the input power in order to achieve extended linear region while maintaining high PAE. The red line in Figure 4.3.a shows one example of the desired Aux PA control policies. Under this policy, the Aux PA setting is gradually increased as the input power increases. The increment happens when the lower setting is more than 1dB lower than the gain obtained in the PBO region in order to maintain high PAE. One can consider the red line in Figure 4.3.a as the "synthesized" gain

response under the Aux PA control. This policy's phase response and PAE are plotted in Figure 4.3.b and Figure 4.3.c respectively.

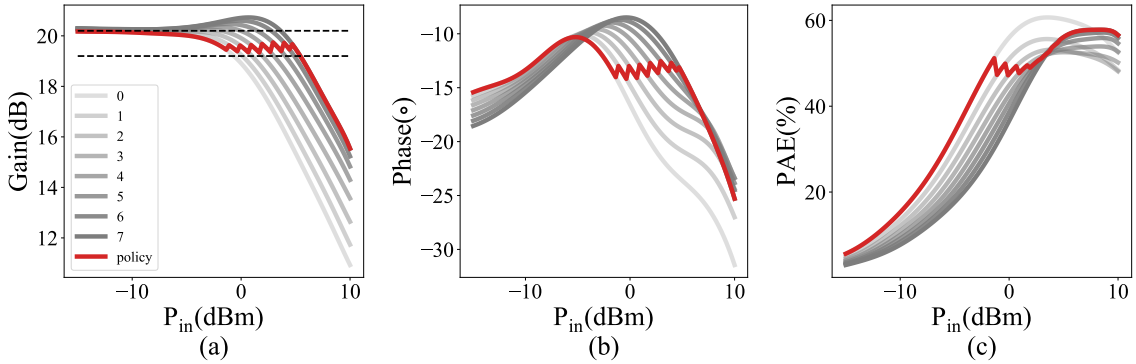


Figure 4.3: Gain(a)/Phase(b) responses and PAE(c) under different Aux PA settings. The VSWR is of standard 50ohm and the Main PA setting is fixed to 6. One example of desired policy which achieves extended linear gain region while maintaining high PAE is plotted as the red line.

Under each Main PA setting, we can obtain a "synthesized" gain response by designing the Aux PA control. The criteria of selecting Main PA setting is, therefore, to choose the one that has the longest linear region in the "synthesized" gain response. Different from the Aux PA control, where the Aux PA setting is selected based on the power of each input sample, the Main PA setting is only changed when an environmental change is detected and the "synthesized" gain response can be improved by switching to another Main PA setting. In other words, in a time-invariant environment, the Main PA setting is fixed, while the Aux PA setting is dynamically adjusted by the control algorithm based on each input's power. One Main PA control algorithm based on stochastic approximation was introduced in our previous publication [78]. The rest of this chapter focuses on designing an efficient Aux PA control algorithm.

4.4 Aux PA Control As Bandit Problems

4.4.1 Closed-Loop Control

Figure 4.2 has demonstrated how environmental variables such as the VSWR affect the PA's performance. In order to adapt the control policy to the environmental changes, we implement a closed-loop control scheme. As shown in Figure 4.1, the control unit observes the PA's input power and selects Main/Aux settings accordingly. The PA's output under the selected setting is sensed and sent back to the control unit. Based on these input/output pairs, the control unit learns the behavior of the PA and optimizes the control policy. The control policy can only be optimized when the control unit has good knowledge about how the PA performs under all settings. Therefore, we apply the idea of exploration and exploitation from reinforcement learning. Each time before sending the control signal to the PA, the control unit chooses between exploitation and exploration. In the exploitation stage, the PA receives the setting actually suggested by the control policy. In the exploration stage, the suggested setting is discarded and some other setting is sent to the PA in order for the control unit to gain broader knowledge about how PA behaves under various settings.

The exploration-exploitation scheme is critical for the PA to maintain optimal performance in a time-variant environment. Environmental changes require the control unit to constantly re-estimate the PA's behavior. Constant control policy update is handled naturally by the exploration-exploitation scheme. The situation when the current control policy is no longer the optimal under the new environment can be identified during the exploration stage, leading to a timely policy adjustment. The balance between exploration and exploitation is affected by how fast the environment changes. When the PA is operating in an environment with known slow time-variance, the exploration rate can be small in order for the PA to follow the actual control policy. However, when the environment is known to change frequently, the exploration rate should be increased to encourage timely performance re-optimization.

4.4.2 Control algorithm based on multi-armed bandit

The Aux PA control can be naturally formulated as a bandit problem. In the multi-armed bandit (MAB) problem, an agent faces a set of actions $\mathcal{A}$, and each action $a \in \mathcal{A}$ returns a stochastic reward r drawn from an unknown distribution. At each time t, the agent chooses one action a_t and receives a reward r_t associated with that action. The goal of the agent is to maximize the cumulative reward in a given time period [119]. Essentially, the agent updates its estimation of the reward distribution based on the received rewards and acts based on the current estimation. Fitting the MAB framework into our Aux PA control, each Aux PA setting can be viewed as one action. As the control unit seeks to achieve the extended linear gain region while maintaining high PAE, we now describe how to design our reward function to reflect this criteria.

The PBO region vs. the learning regions: As shown in Figure 4.3, the gain curves behave differently cross the entire input power range. In the PBO region, the gain curves are flat and nearly stay the same under various Aux PA settings. Therefore, the Aux PA setting is fixed as 0 in this region to achieve the highest PAE. Outisde the PBO region, the power gain curves drop as the input power increases. As a result, the optimal Aux PA setting depends on the input power. We divide this high-input-power region into M "learning regions" and set up one MAB for each region. Figure 4.4 shows the overall MAB arrangement. Within each learning region $\mathcal{Z}^m$, the control unit learns to choose the Aux PA setting that minimizes the gain distortion.

Reward definition: We first estimate a reference power gain $\mathrm{E}[g]$ from the PBO region, where the gain g is viewed as a random variable following some distribution. Our estimation $\tilde{g}$ for this reference gain is calculated using stochastic approximation (SA) [25]. Specifically, when an input falls into the PBO region at time t, we apply the following update:

$$\tilde{g} \leftarrow \tilde{g} + \alpha_0(g_t - \tilde{g}) \,, \tag{4.1}$$

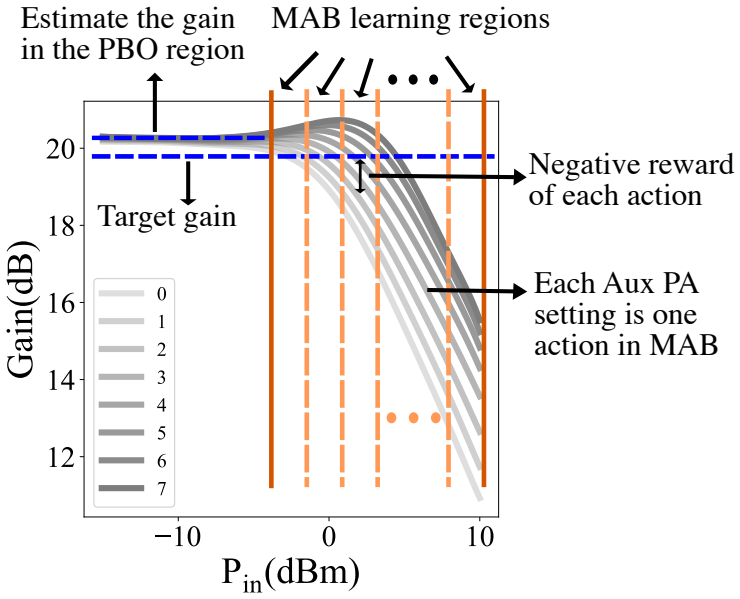


Figure 4.4: The MAB arrangement to achieve the extended linear gain region while maintaining high PAE.

where g_t is the observed power gain at time t and $\alpha_0 \in (0, 1]$ is the SA step size. Essentially, when our observation g_t differs from the estimation $\tilde{g}$, (4.1) makes a small adjustment to the estimation based on the difference. The closer the step size is to 1, the more weight is given to the recent observations. In a stationary setting where the power gain g follows some time-invariant distribution, SA is known to converge to the expectation $\mathrm{E}[g]$ asymptotically when the step size satisfies [25]:

$$\sum_{t=1}^{\infty} \alpha_t = \infty, \quad \sum_{t=1}^{\infty} \alpha_t^2 < \infty \, . \tag{4.2}$$

As our PA is operating in a time-variant environment, we simply set the update size to a fixed value α_0 to constantly adjust the estimation by the most recent observation.

Given a reference gain $\tilde{g}$ estimated from the PBO region, we then design the following reward function for the MAB regions:

$$R_{MAB}^{m}(a) = -\,\mathrm{E}[\mid g^m(a) - \hat{g} \mid] \, , \tag{4.3}$$

where $g^m(a)$ is the observed power gain under Aux PA setting a in region $\mathcal{Z}^m$, and $\hat{g}$ is the reference gain $\tilde{g}$ adjusted by a small constant offset δ: $\hat{g} = \tilde{g} - \delta$. The offset is added to encourage the algorithm to choose a lower Aux PA setting, and hence improving the PAE. The reward measures the negative of the expected absolute difference between the observed gain and the target gain $\hat{g}$.

Updating the reward estimation: With actions and rewards defined above, the control unit treats each learning region as an independent MAB. During operation, the control unit follows the exploration-exploitation scheme. At each time t, an input with power s_t falls into one of the learning regions $\mathcal{Z}^m$. When exploitation is activated, the Aux PA setting suggested by the current reward estimation $a^* = \arg\max_a \hat{R}^m(a)$ is used. When exploration is activated, a randomly selected action different from a^* is used. At each time t, the power gain under the chosen setting a_t is observed: $g_t = g^m(a_t)$, and the estimation for the reward function is updated using SA:

$$\hat{R}^m_{MAB}(a_t) \leftarrow \hat{R}^m_{MAB}(a_t) + \alpha \left(-|g_t - \hat{g}| - \hat{R}^m_{MAB}(a_t) \right), \tag{4.4}$$

where a fixed value $\alpha \in (0, 1]$ is used as the SA step size to adjust the estimation with the most recent observation. The amount of exploration is determined by a fixed number $\epsilon \in (0, 1]$. At every step, the control unit explores with probability ϵ and exploits with probability $1 - \epsilon$, a scheme known as the ϵ-greedy action selection [119]. As a result, the control unit learns one optimal Aux PA setting associated with each MAB learning region. Cross different regions, the control unit adjusts the Aux PA setting according to the input power, achieving the extended linear gain response. The constant exploration and exploitation ensures high performance in a time-variant environment. The complete MAB-based control algorithm is described in Algorithm 4.1.

Testing and results: We first test the performance of the proposed MAB-based algorithm in a time-invariant environment. The VSWR is set to be under a standard 50ohm load. We

Algorithm 4.1 Linear gain control algorithm based on multi-armed bandit

Divided the high input power region into M learning regions $\mathcal{Z}^1, \mathcal{Z}^2, \ldots, \mathcal{Z}^M$.
Initialize the reward estimations $\hat{R}^m_{MAB}(a)$ for each region.
Set stochastic approximation step size α.
Obtain the target gain value $\hat{g}$.
For each time t:
  Observe an input with power s_t that falls into one of the regions $\mathcal{Z}_t = \mathcal{Z}^m$.
  Calculate the best action based on reward estimations:
    $a^* = \arg\max_a \hat{R}^m_{MAB}(a)$.
  With probability $1 - \epsilon$:
    $a_t = a^*$.
  With probability ϵ:
    Randomly choose an action $a_t \neq a^*$.
  Apply action a_t and observe gain g_t.
  Update the reward estimation:
    $\hat{R}^m_{MAB}(a_t) \leftarrow \hat{R}^m_{MAB}(a_t) - \alpha \left(|g_t - \hat{g}| + \hat{R}^m_{MAB}(a_t) \right)$.

set up 25 MAB learning regions with input power from -10dBm to 10dBm. The PBO region has input power lower than -10dBm. The target gain offset δ is set to be 0.5dBm. The exploration ratio ϵ is set as 10%, and the SA step size is set as 0.25 for both MAB update (α) and reference gain estimation (α_0). The reward estimations in all the learning regions are initialized to -1. Figure 4.5.a shows the learned policy after 10,000 samples. The policy stays within 1dB range compared to the PBO gain while holding on to the lower Aux PA settings to achieve higher PAE.

We show the convergence rate of the MAB-based control in Figure 4.5.b. To quantify the convergence, we use the concept of regrets from the reinforcement learning literature. The cumulative regrets C is defined as:

$$C(T) = \sum_{t=0}^{T} r_t^o - r_t = \sum_{t=0}^{T} \mid g_t - \hat{g} \mid - \min_{a_t^o} \mid g(a_t^o) - \hat{g} \mid , \tag{4.5}$$

where r_t^o is the reward would have been received by the optimal policy (if had known) at time t, and r_t is the reward actually received by the learning algorithm. At each time t, the

optimal policy minimizes the absolute difference between the achievable gain and the target gain. We choose 3 different exploration rates: 5%, 15%, and 25%. With each exploration rate, we run 10 trials and calculate the average cumulative regrets. The results are shown in Figure 4.5.b. With large exploration rate, the algorithm learns the optimal policy quickly, but the regrets grow faster afterwards due to the exploration cost. On the contrary, when the exploration rate is small, the algorithm needs more samples to learn the optimal policy, but the regrets grow with a lower rate once the optimal policy is discovered.

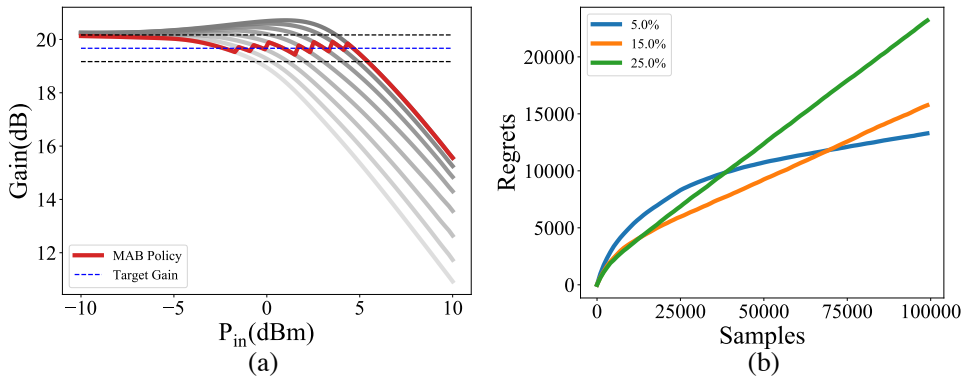


Figure 4.5: Simulation results of MAB-based control algorithm (Algorithm 4.1) in a time-invariant environment. (a) learned policy after 10,000 samples with 10% exploration rate. (b) Cumulative regrets under various exploration rate.

We then test our algorithm in a simulated time-variant environment. In the initialization stage, the VSWR is set to have magnitude of 2 and an angle of 0 degree. The PA is operating in a stable condition with a policy learned by the MAB-based algorithm. We then rotate the VSWR angle for a full circle with 90 degree incremental steps. After each increment, 25,000 samples are sent to the PA. Essentially, the control algorithm has 25,000 samples to learn the new optimal policy and to adjust to the new environment. The algorithm's parameters are the same as in the time-invariant setting with 10% exploration rate. The results after each 25,000 samples are shown in Figure 4.6. The MAB-based control effectively updates the policy to adapt to the simulated environmental changes. The cumulative regrets of the whole simulation, averaged over 10 trials, are shown in Figure 4.7. The sudden slope increments right after every 25,000 samples, marked by the red circles, correspond to the

policy re-adjustment after VSWR angle changes. The regrets significantly increase as the old policy is no longer optimal. Later in each segment, the slope of the regrets decreases and converges to a constant level, indicating the new optimal policy has been learned by the control algorithm.

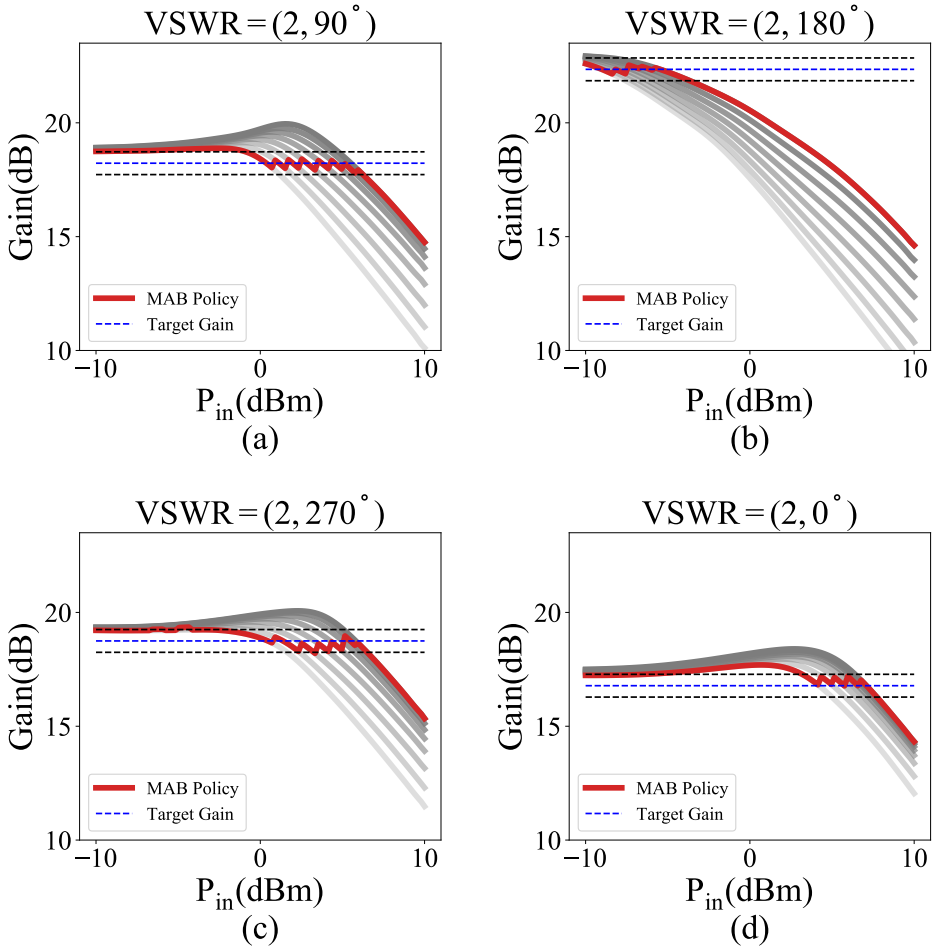


Figure 4.6: Simulation results of the MAB-based control algorithm (Algorithm 4.1) in a time-variant environment. The angle of the VSWR is rotated in the order of 90, 180, 270, and 360 degrees. 25,000 samples are sent to the system after each rotation. The policy learned by the MAB-based control after every 25,000 samples are plotted in (a)-(d).

4.4.3 Control algorithm based on continuum-armed bandit

Although the MAB-based control algorithm is effective with low computational complexity, we can improve the learning efficiency by leveraging more hardware properties. The MAB-

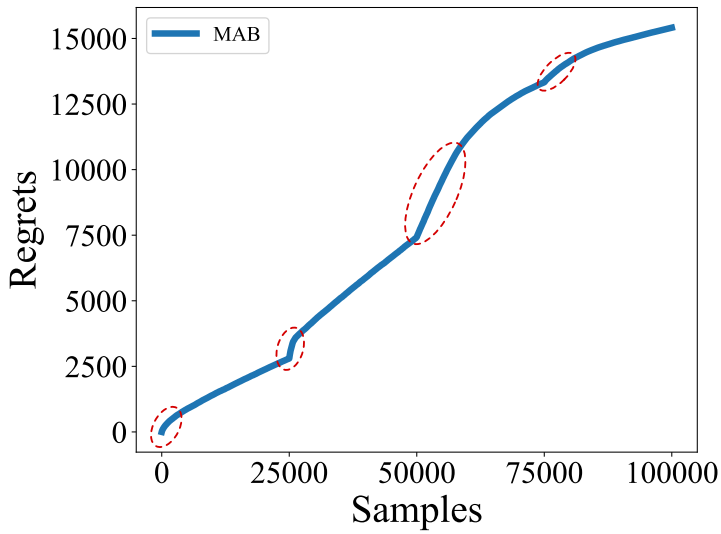


Figure 4.7: Cumulative regrets of the MAB-based control algorithm (Algorithm 4.1) in the time-variant simulation test. The test setting is the same as in Figure 4.6. The result shown is averaged over 10 trials. The red circles show the segments corresponding to the policy re-adjustment after each VSWR angle increment.

based control algorithm treats the Aux PA settings as actions with independent rewards. The reward estimation of a specific action is not updated until the action itself is applied. However, for hardware control problems, the hardware's performance under each setting is often correlated. Figure 4.3 shows that in each learning region the gain monotonically increases as the 3-bit Aux PA setting goes from 0 to 7. This ordering structure allows us to infer the gain of some settings based on their observed neighborhoods before sampling the actions themselves. For example, if we have explored settings 3 and 6 in region $\mathcal{Z}^m$ which has expected gain $G^m(a=3)$ and $G^m(a=6)$ respectively, we can infer that $G^m(a=4)$ and $G^m(a=5)$ should fall between these two values. With our prior knowledge about this ordering structured, we are able extract more information from each observed sample and improve the learning efficiency.

To leverage the correlated Aux PA settings, we propose a function approximation $\hat{G}$ for the power gain in each learning region $\mathcal{Z}^m$:

$$G^m_{CAB}(a) = k^m_1 a + k^m_0 \ , \tag{4.6}$$

where a is the Aux PA setting, and $\boldsymbol{k}^m = [k_1^m, k_0^m]$ is the parameter to be estimated. Within each learning region, the ordering relationship among the Aux PA settings is captured by the a linear model. With this approximated linear gain function, our function approximation of the rewards now measures the difference between the approximated gain and the target gain $\hat{g}$:

$$\hat{R}_{CAB}^m(a) = - \mid \hat{G}_{CAB}^m(a) - \hat{g} \mid = - \mid k_1^m a + k_0^m - \hat{g} \mid \ . \tag{4.7}$$

Using function approximation to exploit the correlations among actions is a common approach when the number of actions is large or when the action is modeled as a continuous variable, a setting known as the continuum-armed bandit (CAB) [120, 121, 122, 123, 23].

Update the reward approximation parameter: The learning procedure of the CAB-based algorithm also follows the exploration-exploitation scheme as in the MAB-based algorithm. With the proposed function approximation, the step of calculating the best action with respect to the current estimation has a closed-form solution:

$$\begin{aligned} a^* &= \left\lfloor \underset{a\in[a_{min},a_{max}]}{\arg\max} \ R_{CAB}^m(a) \right\rceil \\ &= \left\lfloor \underset{a\in[a_{min},a_{max}]}{\arg\max} \ - |k_1^m a + k_0^m - \bar{g}| \right\rceil \\ &= \begin{cases} \min\left(\max\left(\left\lfloor \frac{\hat{g}-k_0^m}{k_1^m} \right\rceil, a_{min}\right), a_{max}\right), \ if \ k_1^m \neq 0 \\ a_{min}, \ if \ k_1^m = 0 \ . \end{cases} \end{aligned} \tag{4.8}$$

The maximizer of $R^m(a)$ is rounded to the nearest valid Aux PA setting bounded by a_{min} and a_{max}.

At each time t, the power gain under the chosen setting a_t is observed: $g_t = g^m(a_t)$, and the estimation of the reward function is updated. First we apply SA to update the estimation of the expected power gain under the chosen setting, denoted by $\hat{g}^m(a_t)$:

$$\hat{g}^m(a_t) \leftarrow \hat{g}^m(a_t) + \alpha \left(g_t - \hat{g}^m(a_t)\right) \ , \tag{4.9}$$

where a fixed value $\alpha \in (0, 1]$ is used as the SA step size to adjust the estimation with the most recent observation. We then update the approximation parameter to minimize the Euclidean distance between the approximated rewards and the estimated expected rewards under each action:

$$\min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(- \mathrm{E}\left[\mid g^m(a) - \hat{g} \mid \right] - \hat{R}^m_{CAB}(a) \right)^2 \tag{4.10}$$

$$= \min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(\mathrm{E}\left[\mid \hat{G}^m_{CAB}(a) - \hat{g} \mid - \mid g^m(a) - \hat{g} \mid \right] \right)^2 \tag{4.11}$$

$$\leq \min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(\mathrm{E}\left[\left| \mid \hat{G}^m_{CAB}(a) - \hat{g} \mid - \mid g^m(a) - \hat{g} \mid \right| \right] \right)^2 \tag{4.12}$$

$$\leq \min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(\mathrm{E}\left[\mid \hat{G}^m_{CAB}(a) - g^m(a) \mid \right] \right)^2 \tag{4.13}$$

$$\approx \min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(\hat{G}^m_{CAB}(a) - \mathrm{E}\left[g^m(a) \right] \right)^2 \tag{4.14}$$

$$\approx \min_{\boldsymbol{k}} \sum_{a=a_{min}}^{a_{max}} \left(\hat{G}^m_{CAB}(a) - \hat{g}^m(a) \right)^2 . \tag{4.15}$$

Given the updated $\hat{g}^m(a)$, we solve the linear square problem (4.15). Therefore, the approximation update step is given by:

$$k_1^m, k_0^m \leftarrow \arg\min_{k_1, k_0} \sum_{a=a_{min}}^{a_{max}} (\hat{g}^m(a) - k_1 a - k_0)^2. \tag{4.16}$$

In practice, the above regression step includes a Tikhonov regularizer to increase the stability and recursive least square (RLS) update is used to improve the computational efficiency [27].The complete CAB-based control algorithm is described in Algorithm 4.2.

Testing and results: To demonstrate the improved learning efficiency of the CAB-based control algorithm, we compare its performance with the MAB-based algorithm's in a simulated time-variant environment. Similar to the test we run for the MAB-based algorithm, the VSWR is set to have magnitude of 2 and an angle of 0 degree at the beginning

Algorithm 4.2 Linear gain control algorithm based on continuum-armed bandit

Divided the high input power region into M learning regions
$\mathcal{Z}^1, \mathcal{Z}^2, \ldots, \mathcal{Z}^M$.
Initialize the the gain estimations $\hat{g}^m(a)$ for each region.
Define reward function in each region as:
$R^m(a) = - \mid k_1^m a + k_0^m - \bar{g} \mid$.
In each region, initialize the reward model's parameters k_1^m, k_0^m.
Set the stochastic approximation step size α.
Obtain the target gain value $\bar{g}$.
For each time t:
  Observe an input with power s_t that falls into one of the regions $\mathcal{Z}_t = \mathcal{Z}^m$.
  Calculate the best action based on reward estimations:
    $a^* = \arg\max_a R^m(a)$.
  With probability $1 - \epsilon$:
    $a_t = a^*$.
  With probability ϵ:
    Randomly choose an action $a_t \neq a^*$.
  Apply action a_t and observe gain g_t.
  Update the gain estimation:
    $\hat{g}^m(a_t) \leftarrow \hat{g}^m(a_t) + \alpha\,(g_t - \hat{g}^m(a_t))$.
  Update the reward function's parameters using regression:
    $k_1^m, k_0^m \leftarrow \arg\min_{k_1,k_0} \sum_{a_{min}}^{a_{max}} (\hat{g}^m(a) - k_1 a - k_0)^2$.

of the simulation. The PA starts from operating in a stable condition with a policy learned by the control algorithm. We then rotate the VSWR angle for a full circle with 90 degree incremental steps. This time, after each increment, 5,000 (instead of 25,000 as for the MAB-based algorithm) samples are sent to the PA. The same set of samples are used for both algorithms. We set up 25 learning regions with input power from -10dBm to 10dBm. The PBO region has input power lower than -10dBm. The target gain offset δ is set to be 0.5dB. The exploration ratio ϵ is set as 10%, and the SA step size is set as 0.25.

The results after each 5,000 samples are shown in Figure 4.8. The upper row shows the policies learned by the MAB-based algorithm. As the number of samples is limited after each VSWR change, the MAB-based algorithm fails to adjust to the new optimal policy due to insufficient exploration of various actions in different learning regions. The lower row in Figure 4.8 shows the policy learned by the CAB-based algorithm. The reward

function approximation significantly improves the algorithm's learning efficiency, and the new optimal policies are successfully learned at the end of each 5000 samples.

Figure 4.9 shows the cumulative regrets of both the MAB-based and the CAB-based algorithms through out the whole simulation. The results as shown are averaged over 10 trials. The improved learning efficiency is evident in this cumulative regrets plot. The regrets of the CAB-based algorithm grow with a much lower rate and the total regrets at the end of the simulation test is only about $2/3$ of that of the MAB-based algorithm.

4.4.4 Control algorithm based on contextual continuum-armed bandit

We can further improve the learning efficiency by leveraging more hardware properties. The MAB-based and the CAB-based algorithm essentially approximates the power gain under each Aux PA setting by a constant within each learning region Z^m and one optimal action is chosen for the region. The entire gain curves are, therefore, coarsely approximated by piecewise constant functions. A better approximation scheme that enables adjustments on the actions within each learning region has the potential to improve the learned policy.

Approximate the reward function: To both leverage the correlated Aux PA settings and allow action fine-tuning within each learning region, we propose a function approximation $\hat{G}^m_{CCAB}$ for the power gain in each learning region $\mathcal{Z}^m$:

$$\hat{G}^m_{CCAB}(s,a) = k_3^m sa + k_2^m s + k_1^m a + k_0^m \ , \tag{4.17}$$

where $s \in \mathcal{Z}^m$ is the input power, a is the Aux PA setting, and $\boldsymbol{k}^m = [k_3^m, k_2^m, k_1^m, k_0^m]$ is the parameter to be estimated. Under a given input power s_0, the ordering relationship among the Aux PA settings is captured by a linear model: $\hat{G}(s = s_0, a) = (k_3^m s_0 + k_1^m)a + k_2^m s_0 + k_0^m$. Under each Aux PA setting, the gain is also approximated by a linear function $\hat{G}(s, a = a_0) = (k_3^m a_0 + k_2^m)s + k_1^m a_0 + k_0^m$. With this approximated bilinear gain function, our estimated reward function now measures the difference between the approximated gain and

the target gain $\hat{g}$:

$$\begin{aligned}\hat{R}^m_{CCAB}(s,a) &= -\mid \hat{G}^m_{CCAB}(s,a) - \hat{g} \mid \\ &= -\mid k^m_3 sa + k^m_2 s + k^m_1 a + k^m_0 - \hat{g} \mid \ .\end{aligned} \tag{4.18}$$

Our new reward function is assumed to depend on an observable state: input power s. The information contained in the observable state helps the control unit make better decision. The control unit seeks to learn a policy mapping from the observable states to the actions that generates the largest rewards. By including the states into the reward function approximation, we formulate the Aux PA control as a contextual bandit [119, 124, 121, 24]. Hence, the subscription, CCAB, in (4.18) stands for contextual continuum-armed bandit.

Update the reward approximation parameters: The learning procedure of the CCAB-based algorithm also follows the exploration-exploitation scheme as in the MAB-based and the CAB-based algorithms. With the proposed bilinear reward function, the step of calculating the best action with respect to the current estimation has a closed-form solution:

$$\begin{aligned} a^* &= \left\lfloor \underset{a\in[a_{min},a_{max}]}{\arg\max}\ \hat{R}^m_{CCAB}(s=s_t,a) \right\rceil \\ &= \begin{cases} \min\left(\max\left(\left\lfloor \frac{\hat{g}-k^m_2 s_t - k^m_0}{k^m_3 s_t + k^m_1} \right\rceil, a_{min}\right), a_{max}\right), \ if\ k^m_3 s_t + k^m_1 \neq 0 \\ a_{min},\ if\ k^m_3 s_t + k^m_1 = 0\ . \end{cases} \end{aligned} \tag{4.19}$$

The maximizer of $\hat{R}^m_{CCAB}(s_t,a)$ is rounded to the nearest valid Aux PA setting bounded by a_{min} and a_{max}.

During the PA's operation, we collect data entries in the form of triple tuples containing the input power, the Aux PA setting, and the corresponding power gain measured at the end of the feedback control loop: $d_i = (s_i, a_i, g_i)$. Given a set of N data entries in a learning region $\mathcal{Z}^m$, we update the approximation parameter to minimize the Euclidean distance

between the approximated rewards and the true rewards:

$$\min_{\boldsymbol{k}} \sum_{i=1}^{N} \left(r_i - \hat{R}^m_{CCAB}(s_i, a_i) \right)^2 \tag{4.20}$$

$$= \min_{\boldsymbol{k}} \sum_{i=1}^{N} \left(\mid \hat{G}^m_{CCAB}(s_i, a_i) - \hat{g} \mid - \mid g_i - \hat{g} \mid \right)^2 \tag{4.21}$$

$$\leq \min_{\boldsymbol{k}} \sum_{i=1}^{N} \left(\hat{G}^m_{CCAB}(s_i, a_i) - g_i \right)^2 . \tag{4.22}$$

The non-linear least square problem (4.21) is upper bounded by the linear least square problem (4.22) which is much easier to solve. Therefore, the approximation update step is given by:

$$\boldsymbol{k}^m \leftarrow \arg\min_{\boldsymbol{k}} \sum_{d_i \in B^m} \left(g_i - k_3 s_i a_i - k_2 s_i - k_1 a_i - k_0 \right)^2 , \tag{4.23}$$

where B^m is a first-in-first-out (FIFO) buffer to hold L most recent observed data within the learning region. The FIFO buffer allows the algorithm to adapt to the time-variant environment and the buffer length L determines the rate of adaption. Formulating the function parameter update as solving least square problems has been widely implemented in previous works on both the continuum-armed bandit [123, 23] and the contextual bandit [24].

In practice, the above regression step includes a Tikhonov regularizer to increase the stability and recursive least square (RLS) update is used to improve the computational efficiency [27]. Compared to the MAB-based algorithm, the CCAB-base algorithm has additional cost in both computation and memory. The computational complexity is mainly increased by the least square step. The RLS update with one addition sample avoids matrix inversion but still has complexity $\mathcal{O}\left(N_k^2\right)$, where $N_k = 4$ in our case is the number of parameters to be estimated. The memory requirement is mainly increased by the implementation of FIFO buffers which whose cost is $\mathcal{O}\left(L\right)$. The least square problem (4.22) can also be solved using iterative methods such as stochastic gradient descent (SGD). Although

SGD reduces both the computational and memory requirements at each step, it in general requires more steps to reach a satisfactory result. Therefore, extracting more information from each sample comes with the cost of addition computation and memory requirements. This trade-off ought to be considered when designing PA systems for different applications. The CCAB-based linear gain control is summarized in Algorithm 4.3.

Algorithm 4.3 Linear gain control algorithm based on contextual continuum-armed bandit

Divided the high input power region into M learning regions $\mathcal{Z}^1, \mathcal{Z}^2, \ldots, \mathcal{Z}^M$.
Define the reward approximation function in each region as:
$R^m_{CCAB}(s,a) = -\mid k^m_3 sa + k^m_2 s + k^m_1 a + k^m_0 - \hat{g} \mid$.
In each region, initialize the reward model's parameters $\boldsymbol{k}^m$.
Set up one FIFO buffer B^m of length L in each learning region.
Obtain the target gain value $\hat{g}$.
For each time t:
Observe an input with power s_t that falls into one of the regions $\mathcal{Z}_t = \mathcal{Z}^m$.
Calculate the best action based on the reward approximation function:

$$a^* = \left\lfloor \underset{a\in[a_{min},a_{max}]}{\arg\max} \; R^m_{CCAB}(s_t,a) \right\rceil.$$

With probability $1-\epsilon$:
$a_t = a^*$.
With probability ϵ:
Randomly choose an action $a_t \neq a^*$.
Apply action a_t and observe gain g_t.
Add $d_t = (s_t, a_t, g_t)$ into FIFO buffer B^m.
Remove the earliest data in B^m if the length of B^m is larger than L.
Update model parameters using regression:

$$\boldsymbol{k}^m \leftarrow \underset{\boldsymbol{k}}{\arg\min} \sum_{d_i \in B^m} (g_i - k_3 s_i a_i - k_2 s_i - k_1 a_i - k_0)^2.$$

Testing and results: To demonstrate the improved learning efficiency of the CCAB-based control algorithm, we compare its performance with the MAB-based algorithm's in a simulated time-variant environment. The simulation settings are similar to the ones used for the MAB-based algorithm described in Section 4.4.2. Several changes are introduced in this test. The length of each FIFO buffer B^m is set to be 50. The number of samples after each VSWR angle increment is set to 5,000, which is significantly fewer than the number of samples used in the testing for the MAB-based algorithm. This reduction is introduced

to show the improved learning efficiency of the CCAB-based algorithm. The same set of samples are used for the two algorithms during each trial.

The results after each 5,000 samples are shown in Figure 4.10. The upper row shows the policies learned by the MAB-based algorithm. As the number of samples is largely reduced after each VSWR change, the MAB-based algorithm fails to adjust to the new optimal policy due to insufficient exploration of various actions in different learning regions. The lower row in Figure 4.10 shows the policy learned by the CCAB-based algorithm. The reward approximation function significantly improves the algorithm's learning efficiency, and the new optimal policies are successfully learned by the end of each 5000 samples. The improved learning efficiency is also evident in the cumulative regrets plot in Figure 4.11. The cumulative regrets as shown are averaged over 10 trials. The regrets of the CCAB-based algorithm grow with a much lower rate and the total regrets at the end of the simulation test is only about $1/3$ of that of the MAB-based algorithm.

4.4.5 Control algorithm based on the actor-critic framework

The CCAB-based control algorithm applies function approximation to the rewards within each learning region and exploits the correlation among actions. However, there is no relationship assumed cross different learning regions, and the learning processes within each region are independent. As suggested in Figure 4.3.a, the gain curve under each Aux PA control setting is smooth and has a clear ordering structure. The desired policy gradually increases the Aux PA setting as the input power goes up. In other words, if the input power of region $\mathcal{Z}^{mid}$ is between the input power of region $\mathcal{Z}^{low}$ and $\mathcal{Z}^{high}$, then the control parameters a of each region should also satisfy $a^{low} \leq a^{mid} \leq a^{high}$. We can, therefore, infer the optimal action of a learning region before exploring its actions as long as its neighborhood is well explored. In this way, the learning efficiency can be further improved.

Policy function: To exploit the relationship among the learning regions, we construct a

piecewise linear policy function:

$$\Pi_{c_0,c_1}(s) = \max\left(\min(c_1 s + c_0, a_{max}), a_{min}\right) \ . \tag{4.24}$$

A policy function is a mapping from states (input power) to actions (Aux PA settings). The output of the function is continuous which will be rounded to the nearest integer as a valid Aux PA setting: $a^* = \lfloor \Pi_{\boldsymbol{c}}(s) \rceil$. The policy function has two parameters: c_0 and c_1, where c_1 controls how fast the Aux PA control parameter increases with the input power, and c_0 controls the positions of the upper and lower saturation points. One example of the policy function is plotted in Figure 4.12. The blue lines in the plot correspond to the policy shown earlier in Figure 4.3. This policy can be approximated by the piecewise linear policy function shown as the red dashed line.

Policy gradient: With the policy function defined in (4.24), learning the optimal policy is equivalent of searching for the policy parameter $\boldsymbol{c} = [c_0, c_1]$ that leads to the highest expected reward. By combining the policy function with the reward function approximation in (4.18), we can write the following equation as the approximated expected reward of a policy parameterized by $\boldsymbol{c}$:

$$\begin{aligned} \hat{R}_{AC}(\boldsymbol{c}) =& \mathrm{E}\left[\sum_{m=1}^{M} \mathbf{1}_{\mathcal{Z}^m}(s)\hat{R}^m_{AC}(s,\boldsymbol{c})\right] \\ =& -\mathrm{E}\left[\sum_{m=1}^{M} \mathbf{1}_{\mathcal{Z}^m}(s) \mid k_3^m s\Pi_{\boldsymbol{c}}(s) + k_2^m s + k_1^m \Pi_{\boldsymbol{c}}(s) + k_0^m - \hat{g} \mid \right], \end{aligned} \tag{4.25}$$

where $\mathbf{1}_{\mathcal{Z}^m}(s)$ is the indicator function that shows which learning region the input power s falls into, and the expectation is taken over the input power which is assumed to follow some distribution. This expected reward if controlled by two sets of parameters: the bilinear reward function parameter $\boldsymbol{k}$ and the policy parameter $\boldsymbol{c}$. Given the current estimation of the reward function, we can maximize the reward by applying gradient ascent on $\boldsymbol{c}$. The gradient

of the expected reward (4.25) with respect to the policy parameters can be calculated as:

$$\sigma^m = \text{sgn}(k_3^m s\Pi_{\boldsymbol{c}}(s) + k_2^m s + k_1^m \Pi_{\boldsymbol{c}}(s) + k_0^m - \hat{g}) \tag{4.26}$$

$$\frac{\partial \hat{R}_{AC}^m(s,\boldsymbol{c})}{\partial c_0} = \begin{cases} -\sigma^m(k_3^m s + k_1^m) & a_{min} \leq c_1 s + c_0 \leq a_{max} \\ 0 & elsewhere \end{cases} \tag{4.27}$$

$$\frac{\partial \hat{R}_{AC}^m(s,\boldsymbol{c})}{\partial c_1} = \begin{cases} -\sigma^m s(k_3^m s + k_1^m) & a_{min} \leq c_1 s + c_0 \leq a_{max} \\ 0 & elsewhere \end{cases} \tag{4.28}$$

$$\frac{\partial \hat{R}_{AC}(\boldsymbol{c})}{\partial c_0} = \text{E}\left[\sum_{m=1}^{M} \mathbf{1}_{\mathcal{Z}^m}(s) \frac{\partial \hat{R}_{AC}^m(s,\boldsymbol{c})}{\partial c_0}\right] \tag{4.29}$$

$$\frac{\partial \hat{R}_{AC}(\boldsymbol{c})}{\partial c_1} = \text{E}\left[\sum_{m=1}^{M} \mathbf{1}_{\mathcal{Z}^m}(s) \frac{\partial \hat{R}_{AC}^m(s,\boldsymbol{c})}{\partial c_1}\right] . \tag{4.30}$$

Evaluating the exact gradients in (4.29, 4.30) is inapplicable when the PA is under operation. Hence, we apply the stochastic gradient (SG) ascent [125] algorithm with the gradient estimation given by (4.27, 4.28). Specifically, at each time step t with the input power s_t falling into the learning region $\mathcal{Z}^m$, we update the policy parameter:

$$\boldsymbol{c} \leftarrow \boldsymbol{c} + \beta \frac{\partial \hat{R}_{AC}^m(s_t,\boldsymbol{c})}{\partial \boldsymbol{c}} , \tag{4.31}$$

where β is the SG step size. In practice, running the vanilla SG algorithm on the policy function defined in (4.24) may encounter the gradient diminishing problem in the two saturation regions of the policy function. This problem can be solved by specifying a polygonal feasible region for $\boldsymbol{c}$ and projecting the updated $\boldsymbol{c}$ onto this feasible region after each SG update, a method known as the projected gradient ascent [126, 127, 128]. Updating policy based on forming gradients with respect to the policy parameters is known as the policy gradient method [129, 130] in the reinforcement learning literature.

As we update the policy parameters based on the estimated reward function, it is crucial for our reward function to closely approximate the true rewards. The update for the reward

function parameter $\boldsymbol{k}$ stays the same as described in Section 4.4.4: One FIFO buffer B^m is constructed for each learning region to store observed input power, Aux PA setting, and the corresponding power gain. $\boldsymbol{k}^m$ is updated by running regression on all the data points in B^m.

Actor-Critc: In this new algorithm, we have two parameterized function approximations, one for the policy and one for the reward. This approach is known as the actor-critic (AC) framework in reinforcement learning [131, 132, 133]. The actor, which is the policy function, selects actions based on the policy parameters $\boldsymbol{c}$. The critic, which is the reward function, seeks to closely approximate the rewards generated by these actions. Based on the estimated reward function, critic provides feedback to the actor on how to improve the actions in order to increase rewards. In our case, the feedback is in the form of policy gradient. During the whole learning process, the critic and the actor are updated alternatively to improve reward function approximation and the policy function respectively. The bilinear reward function approximation in each learning region can be viewed as an individual critic. Cross all learning region, these critics work together to improve the policy function. Here, we approximate the complex-shaped overall rewards by multiple simple bilinear models. The policy function can also be viewed as a regularizer on all the local critics, which improves the consistency of the overall policy.

Similar to the other control algorithms we have discussed so far, the AC-based algorithm follows the exploration-exploitation scheme. At each time t, the reward function (critic) parameter $\boldsymbol{k}$ is updated. The policy function (actor) parameter $\boldsymbol{c}$, on the other hand, is only updated when exploitation is activated. The AC-based algorithm converges under time-invariant settings. Notice that the update of the reward function parameter $\boldsymbol{k}$ does not depend on the policy parameter $\boldsymbol{c}$. As $t \to \infty$ and $L \to \infty$, $\boldsymbol{k}$ converges because of the linear regression update. As $\boldsymbol{k}$ converges, $\boldsymbol{c}$ converges under the standard SG convergence conditions [25, 125].

Experience replay: The actor update step performs gradient ascent on the policy func-

tion. Although the computational cost of the update is low, a large number of SG steps may be needed for the policy function to improve. When there is only one SG step within each learning step, it can take a long time to learn the optimal policy. One way to improve the learning rate is to perform more SG updates within some of the learning steps. For example, we can perform one step policy update for the majority of the time. Then every T_e time period, we update the policy function to be the optimal with respect to the current reward function estimation. Since we already have the previous input power stored in the FIFO buffers, we can use these stored values to approximate the expected reward:

$$\hat{R}_{ER}(\boldsymbol{c}) = \frac{1}{N_B} \sum_{m=1}^{M} \sum_{s_i \in B^m} |k_3^m s_i \Pi_{\boldsymbol{c}}(s_i) + k_2^m s_i + k_1^m \Pi_{\boldsymbol{c}}(s_i) + k_0^m - \hat{g}\ | \,, \tag{4.32}$$

where N_B is the total number of samples stored in the buffers. We update the policy parameter to the maximizer of this approximated expected reward:

$$\boldsymbol{c}_{ER} = \arg\min_{\boldsymbol{c}} \hat{R}_{ER}(\boldsymbol{c}). \tag{4.33}$$

The optimization problem is solved by running the projected gradient ascent algorithm. This update step re-uses the historical data stored in the buffer. This learning method is known as "experience replay" (ER) in the reinforcement learning literature [134, 135, 136].

An alternative way to perform ER is to enforce the policy function to match the best actions suggested by the current reward function estimation. Given reward function parameters $\boldsymbol{k}$, the optimal action under state s is given by (4.19). Therefore, we first calculate the optimal action corresponding to each input power stored in the buffer, and then regress the policy function on these action data points:

$$\boldsymbol{c}_{ER} = \arg\min_{\boldsymbol{c}} \sum_{m=1}^{M} \sum_{s_i \in B^m} \left(\arg\max_{a} R_{CCAB}^m(s_i, a) - \Pi_{\boldsymbol{c}}(s_i) \right)^2 . \tag{4.34}$$

This optimization problem is also solved using the projected gradient descent algorithm. In

practice, we observe that the two ER methods have similar performance. When the number of samples in the buffers is large, we randomly sample a mini-batch to evaluate the gradient at each SG step. The policy function parameter $\boldsymbol{c}$ before the ER update is used as the starting point of the optimization problem. As the critic update is essentially the same as in the CAAB-based algorithm, the main contributor to the additional computational complexity is the ER step which contains internal iterations. The AC-based algorithm with experience replay is summarized in Algorithm 4.4.

Testing and results: We demonstrate the further improved learning efficiency of the AC-based control algorithm by comparing its performance with the CCAB-based algorithm's in a simulated time-variant environment. The simulation settings are similar to the ones in the two aforementioned tests. The length of each FIFO buffer B^m is set to be 50 for both algorithms. To show the improvement in the learning efficiency distinctly, we further reduce the number of samples after each VSWR angle increment to 1,500. The exploration rate is reduced from 10% as in the previous tests to 5%. The step size for the actor update is set to be 0.05 and the experience replay is performed every 10 samples.

The results after each 1,500 samples are shown in Figure 4.13. The upper row shows the policies learned by the CCAB-based algorithm. As the number of samples after each VSWR change is further reduced, the CCAB-based algorithm struggles to adjust to the new optimal policy. Although the optimal actions are learned in the well-explored learning regions, the inferior action choices in the other regions lead to an inconsistent policy overall. The lower row in Figure 4.13 shows the policy learned by the AC-based algorithm. The piecewise linear policy function acts as a regularizer and significantly improves the consistency of the policy. The new optimal policies are successfully learned by the end of each 1,500 samples. Therefore, by imposing a policy function that leverages the correlation among learning regions, the AC-based control improves the learning efficiency. Figure 4.14 quantifies this improvement on the cumulative regrets plot which averages the results over 10 trials. As the simulation test progresses, the regrets of the AC-based algorithm grows with a lower rate.

Algorithm 4.4 Linear gain control algorithm based on the actor-critic framework with experience replay

Set up ACTOR
  Initialize the parameters c_0, c_1 of the policy function:
  $\Pi_{\boldsymbol{c}}(s) = \max\left(\min(c_1 s + c_0, a_{max}), a_{min}\right)$.
  Set stochastic gradient ascent step size β.
Set up CRITIC
  Divided the high input power region into M learning
  regions $\mathcal{Z}^1, \mathcal{Z}^2, \ldots, \mathcal{Z}^M$.
  Define the reward function in each learning region as:
  $\hat{R}^m_{AC}(s, \boldsymbol{c}) = -\mid k^m_3 s\Pi_{\boldsymbol{c}}(s) + k^m_2 s + k^m_1 \Pi_{\boldsymbol{c}}(s) + k^m_0 - \hat{g} \mid$.
  In each region, initialize the reward model's parameters $\boldsymbol{k}^m$.
  Set up one FIFO buffer of length L in each learning region.
  Obtain the target gain value $\hat{g}$.
Set experience replay interval time T_e.
For each time t:
  Observe an input with power s_t that falls into one of the regions $\mathcal{Z}_t = \mathcal{Z}^m$.
  Calculate the action suggested by the ACTOR:
  $a^* = \lfloor \Pi_{\boldsymbol{c}}(s_t) \rceil$.
  With probability $1 - \epsilon$:
  $a_t = a^*$.
  With probability ϵ:
  Randomly choose an action $a_t \neq a^*$.
  Apply action a_t and observe gain g_t.
  Update CRITIC:
    Add $d_t = (s_t, a_t, g_t)$ into FIFO buffer B^m.
    Remove the earliest data in B^m if the length of B^m is larger than L.
    Update model parameters using regression:
    $\boldsymbol{k}^m \leftarrow \arg\min_{\boldsymbol{k}} \sum_{d_i \in B^m} (g_i - k_3 s_i a_i - k_2 s_i - k_1 a_i - k_0)^2$.
  Update ACTOR **if** $a_t == a^*$:
    $\boldsymbol{c} \leftarrow \boldsymbol{c} + \beta \frac{\partial \hat{R}^m_{AC}(s_t, \boldsymbol{c})}{\partial \boldsymbol{c}}$.
  Perform experience replay **if** $t \ \% \ T_e == 0$:
    $\boldsymbol{c} \leftarrow \arg\min_{\boldsymbol{c}} \sum_{m=1}^{M} \sum_{s_i \in B^m} \mid k^m_3 s_i \Pi_{\boldsymbol{c}}(s_i) + k^m_2 s_i + k^m_1 \Pi_{\boldsymbol{c}}(s_i) + k^m_0 - \hat{g} \mid$,
    or
    $\boldsymbol{c} \leftarrow \arg\min_{\boldsymbol{c}} \sum_{m=1}^{M} \sum_{s_i \in B^m} \left(\arg\max_{a} R^m_{CCAB}(s_i, a) - \Pi_{\boldsymbol{c}}(s_i)\right)^2$.

4.5 Chapter summary

We have proposed 4 PA control algorithms for the Doherty PA to achieve extended linear gain region while maintaining high PAE. The Aux PA settings are dynamically adjusted

based on the input power. The MAB-based algorithm divides the whole input power range into several learning regions and learns one best action for each region. Learning efficiency can be significantly improved by incorporating the prior knowledge about the PA. The CAB-based algorithm leverages the correlations among the Aux PA settings. Built upon the CAB-based algorithm, the CCAB-based algorithm approximates the reward function by a bilinear model to allow more flexible control policies within learning regions. Policy function further leverages the correlations among these learning regions and improves the consistency of the overall policy. In general, there is a trade-off between the learning efficiency and the computational/memory complexity. This trade-off ought to be considered when designing PA systems for different applications.

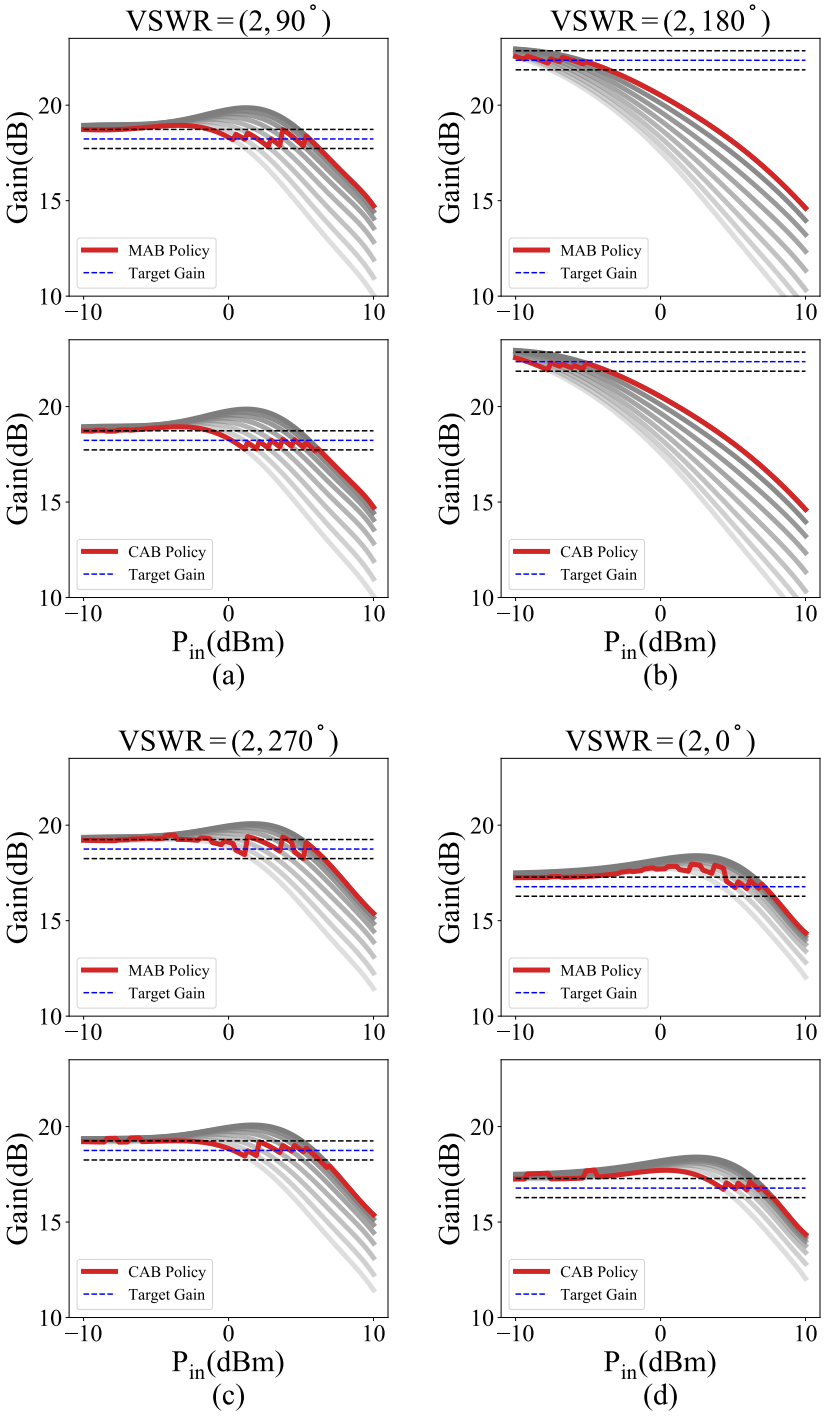


Figure 4.8: Performance comparison between the CAB-based control (Algorithm 4.2) and the MAB-based control (Algorithm 4.1) in a simulated time-variant environment. The angle of the VSWR is rotated in the order of 90, 180, 270, and 360 degrees. 5,000 samples are sent to the system after each rotation. The learned policies after every 5,000 samples are plotted in (a)-(d), with the one learned by the MAB-based algorithm on the top and the one learned by the CAB-based algorithm at the bottom.

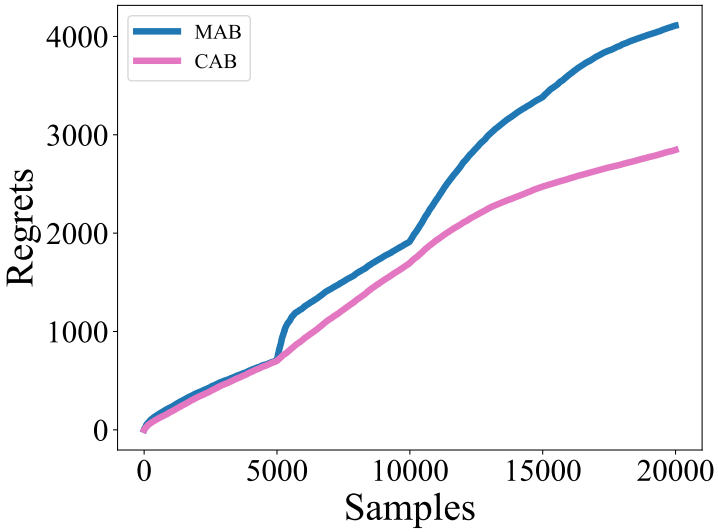


Figure 4.9: Cumulative regrets of the CAB-based (Algorithm 4.2) control algorithm in the time-variant simulation test. The test setting is the same as in Figure 4.8. The result shown is averaged over 10 trials. The regrets of the MAB-based (Algorithm 4.1) control algorithm is also plotted as a comparison.

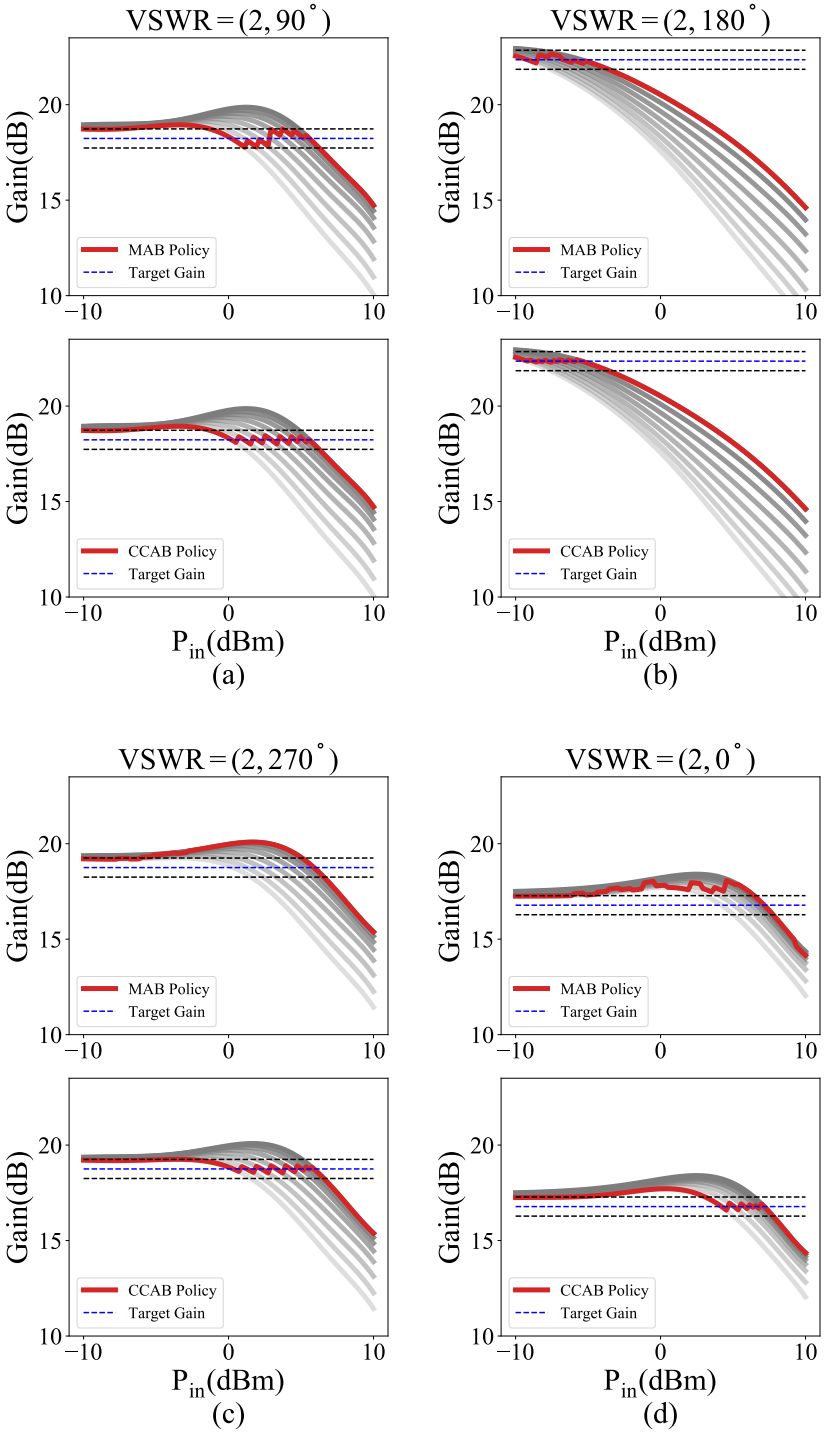


Figure 4.10: Performance comparison between the CCAB-based control and the MAB-based control in a simulated time-variant environment. The angle of the VSWR is rotated in the order of 90, 180, 270, and 360 degrees. 5,000 samples are sent to the system after each rotation. The learned policies after every 5,000 samples are plotted in (a)-(d), with the one learned by the MAB-based algorithm on the top and the one learned by the CCAB-based algorithm at the bottom.

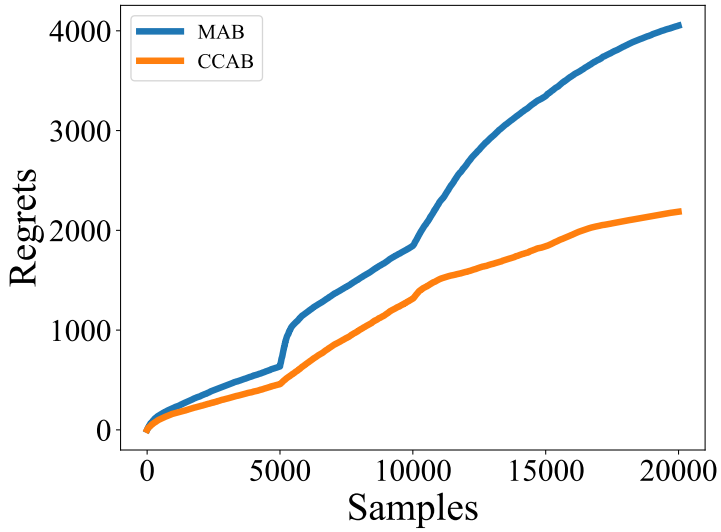


Figure 4.11: Cumulative regrets of the CCAB-based control algorithm in the time-variant simulation test. The test settings are the same as in Figure 4.10. The result shown is averaged over 10 trials. The regrets of the MAB-based control algorithm are also plotted as a comparison.

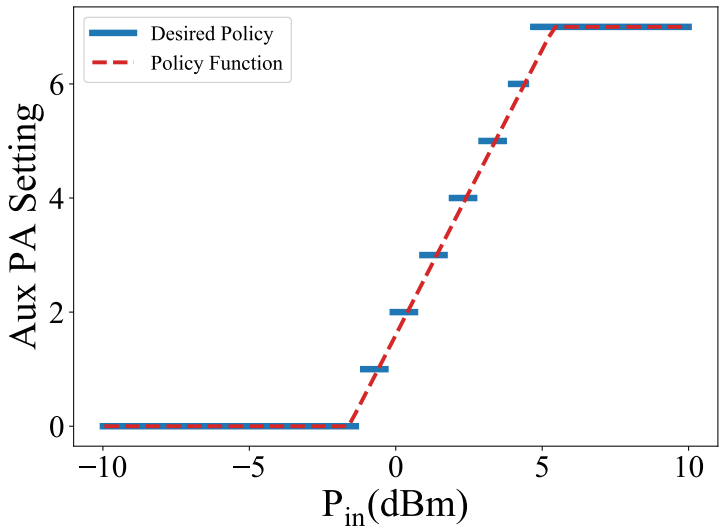


Figure 4.12: Policy function plot. The blue lines correspond to the policy in Figure 4.3. The red dashed line shows the approximation of this policy using the piecewise linear policy function defined in (4.24).

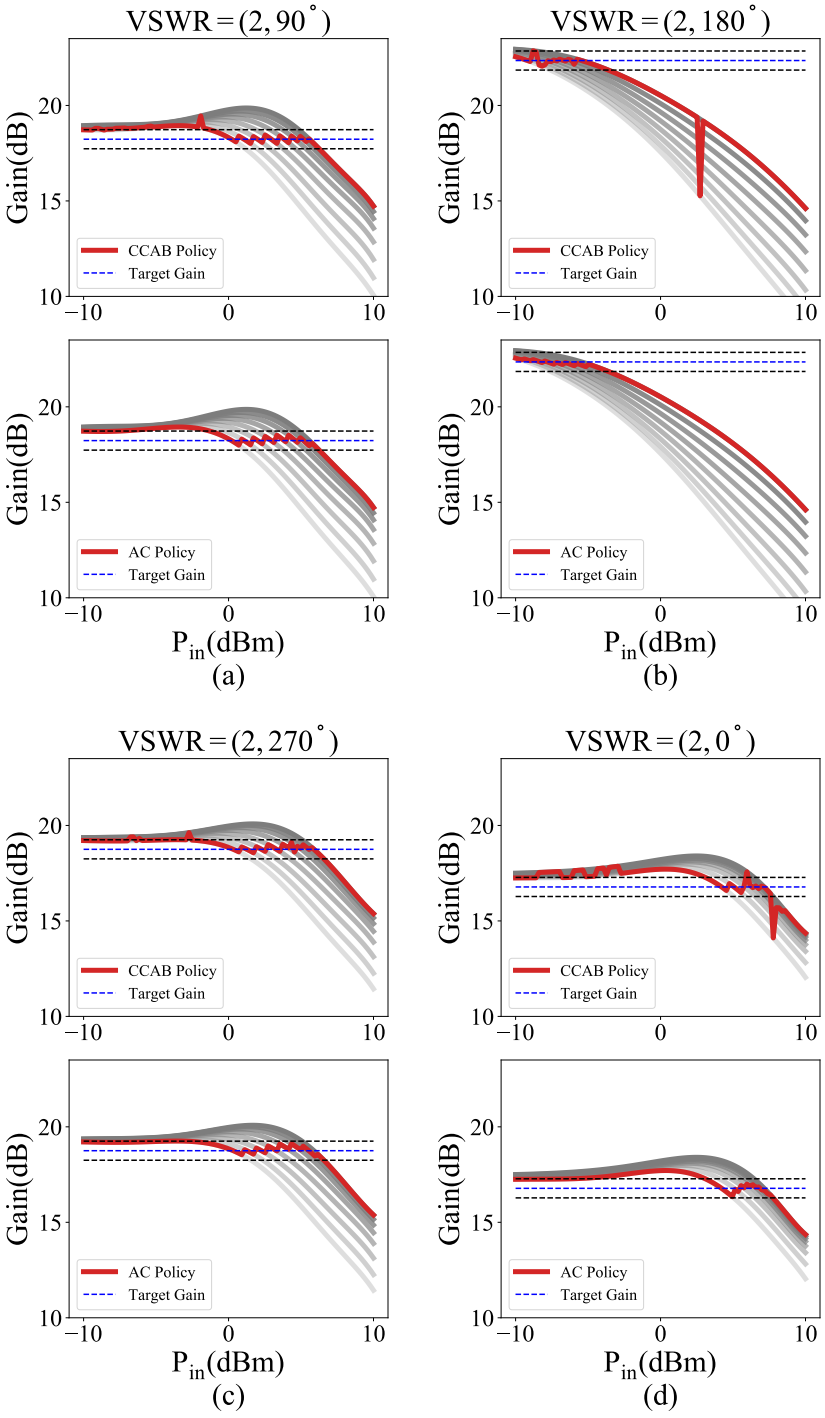


Figure 4.13: Simulation results of the AC-based control algorithm in a time-variant environment. The angle of the VSWR is rotated in the order of 90, 180, 270, and 360 degrees. 1,500 samples are sent to the system after each rotation. The learned policies after every 1,500 samples are plotted in (a)-(d), with the one learned by the CCAB-based algorithm on the top and the one learned by the AC-based algorithm at the bottom.

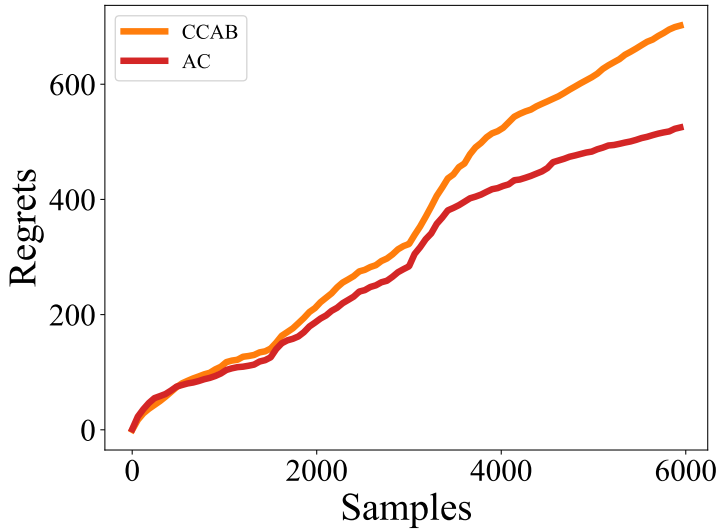


Figure 4.14: Cumulative regrets of the AC-based control algorithm in the time-variant simulation test. The test setting are the same as in Figure 4.13. The result shown is averaged over 10 trials. The regrets of the CCAB-based control algorithm is also plotted as a comparison.

CHAPTER 5
CONCLUSION

We have finished our discussions about the three aims we proposed at the beginning of this thesis. For the first aim, *to design data-driven signal processing algorithms which require fewer measurements taken from the sensor front-end*, we developed a compressive sensing recovery algorithm. We used a GAN with structured latent variable space to capture the prior information of the signals. The recovery algorithm follows the ADMM optimization steps, with one sub-routine being accelerated by another neural network. We achieved fast recovery speed and high recovery quality based on fewer compressed measurements.

For the second aim, *to develop algorithm-hardware co-design techniques for hardware that performs specific machine learning tasks*, we proposed a motion gesture recognition algorithm which works directly with video frames captured using compressive sensing techniques. The motion parameters are estimated in the compressed domain, and the estimation algorithm is implemented in the mixed-signal circuits. We also improved the computational and memory efficiency of existing gesture classifiers.

For the third aim, *to design adaptive hardware control algorithms*, we developed multiple Doherty PA control algorithms based on the bandit frameworks. By incorporating the prior information about the Doherty PA's characteristics into our algorithm design, we improved learning efficiency and enabled robust adaptive operations of the PA in time-variant environments.

By the time of writing this thesis, there are still some open questions to our interest. First, the bandit framework in the PA control project assumes the PA to have no memory effect. The control scheme becomes much more complicated when the memory effect is considered. In that case, a good direction of research is to explore reinforcement learning algorithms based on MDP.

Furthermore, the actor-critic-based control algorithm poses an interesting theoretical question. We can abstract the following setting from our PA control application: Given a sequence of K bandits $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_K$, at each step a randomly chosen bandit $\mathcal{B}_k$ is presented in front of the player. The player knows the index of the bandit and each bandit has the equal probability of being selected. We further assume that the optimal action of each bandit $a^*(k)$ is Lipschitz continuous with respect to the index k. Then the theoretical question we would like to answer is what the player's optimal policy is and what the regrets bound is.

As research areas such as machine learning, compressive sensing, IoT, and 5G telecommunication continue to attract much attention, we believe the interdisciplinary developments among these fields will relate machine learning to the hardware's performance improvement tighter then ever before.

REFERENCES

[1] R. Baraniuk and P. Steeghs, "Compressive radar imaging," in *Radar Conference, 2007 IEEE*, IEEE, 2007, pp. 128–133.

[2] D. L. Donoho, "Compressed sensing," *IEEE Transactions on information theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[3] J. Romberg, "Compressive sensing by random convolution," *SIAM Journal on Imaging Sciences*, vol. 2, no. 4, pp. 1098–1128, 2009.

[4] S. Xu, A. Amaravati, J. Romberg, and A. Raychowdhury, "Appearance-based gesture recognition in the compressed domain," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, pp. 1722–1726.

[5] A. Anvesha, S. Xu, N. Cao, J. Romberg, and A. Raychowdhury, "A light-powered, always-on, smart camera with compressed domain gesture detection," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ACM, 2016, pp. 118–123.

[6] A Anvesha, S. Xu, N. Cao, J. Romberg, and A. Raychowdhury, "A light-powered smart camera with compressed domain gesture detection," *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[7] A. Amaravati, S. Xu, J. Romberg, and A. Raychowdhury, "A 65nm compressive-sensing time-based adc with embedded classification and inl-aware training for arrhythmia detection," in *Biomedical Circuits and Systems Conference (BioCAS), 2017 IEEE*, IEEE, 2017, pp. 1–4.

[8] ——, "A 130 nm 165 nj/frame compressed-domain smashed-filter-based mixed-signal classifier for "in-sensor" analytics in smart cameras," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 3, pp. 296–300, 2018.

[9] R. Baraniuk, M. Davenport, R. DeVore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constructive Approximation*, vol. 28, no. 3, pp. 253–263, 2008.

[10] E. J. Candès and M. B. Wakin, "An introduction to compressive sampling," *IEEE signal processing magazine*, vol. 25, no. 2, pp. 21–30, 2008.

[11] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of computational mathematics*, vol. 9, no. 1, pp. 51–77, 2009.

[12] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2013.

[13] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[14] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[15] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, 2016, pp. 2172–2180.

[16] L. Chongxuan, T. Xu, J. Zhu, and B. Zhang, "Triple generative adversarial nets," in *Advances in neural information processing systems*, 2017, pp. 4088–4098.

[17] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. G. Baraniuk, "The smashed filter for compressive classification and target recognition," in *Computational Imaging V*, International Society for Optics and Photonics, vol. 6498, 2007, 64980H.

[18] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," in *International Conference on Machine Learning*, 2017, pp. 537–546.

[19] M. W. Mahoney *et al.*, "Randomized algorithms for matrices and data," *Foundations and Trends® in Machine Learning*, vol. 3, no. 2, pp. 123–224, 2011.

[20] D. P. Woodruff *et al.*, "Sketching as a tool for numerical linear algebra," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 1–2, pp. 1–157, 2014.

[21] C. J.C. H. Watkins, "Learning from delayed rewards," 1989.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[23] P. Rusmevichientong and J. N. Tsitsiklis, "Linearly parameterized bandits," *Mathematics of Operations Research*, vol. 35, no. 2, pp. 395–411, 2010.

[24] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 208–214.

[25] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009, vol. 48.

[26] A. H. Sayed, *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[27] M. H. Hayes, *Statistical digital signal processing and modeling*. John Wiley & Sons, 2009.

[28] D. E. Koulouriotis and A Xanthopoulos, "Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems," *Applied Mathematics and Computation*, vol. 196, no. 2, pp. 913–922, 2008.

[29] O. Besbes, Y. Gur, and A. Zeevi, "Stochastic multi-armed-bandit problem with non-stationary rewards," in *Advances in neural information processing systems*, 2014, pp. 199–207.

[30] ——, "Optimal exploration–exploitation in a multi-armed bandit problem with non-stationary rewards," *Stochastic Systems*, vol. 9, no. 4, pp. 319–337, 2019.

[31] M. Chen, J. Silva, J. Paisley, C. Wang, D. Dunson, and L. Carin, "Compressive sensing on manifolds using a nonparametric mixture of factor analyzers: Algorithm and performance bounds," *IEEE Transactions on Signal Processing*, vol. 58, no. 12, pp. 6140–6155, 2010.

[32] M. Kabkab, P. Samangouei, and R. Chellappa, "Task-aware compressed sensing with generative adversarial networks," *arXiv preprint arXiv:1802.01284*, 2018.

[33] V. Shah and C. Hegde, "Solving linear inverse problems using gan priors: An algorithm with provable guarantees," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 4609–4613.

[34] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on information theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[35] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, *et al.*, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[36] N. Parikh, S. Boyd, *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[37] D. L. Donoho, A. Maleki, and A. Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18 914–18 919, 2009.

[38] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis," in *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[39] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "Bm3d-amp: A new image recovery algorithm based on bm3d denoising," in *Image Processing (ICIP), 2015 IEEE International Conference on*, IEEE, 2015, pp. 3116–3120.

[40] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, 2012, pp. 341–349.

[41] H. C. Burger, C. J. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d?" In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 2392–2399.

[42] J. Rick Chang, C.-L. Li, B. Poczos, B. Vijaya Kumar, and A. C. Sankaranarayanan, "One network to solve them all–solving linear inverse problems using deep projection models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5888–5897.

[43] K. Fan, Q. Wei, L. Carin, and K. A. Heller, "An inner-loop free solution to inverse problems using deep neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 2370–2380.

[44] J. Sun, H. Li, Z. Xu, *et al.*, "Deep admm-net for compressive sensing mri," in *Advances in neural information processing systems*, 2016, pp. 10–18.

[45] H. K. Aggarwal, M. P. Mani, and M. Jacob, "Modl: Model-based deep learning architecture for inverse problems," *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 394–405, 2018.

[46] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned d-amp: Principled neural network based compressive image recovery," in *Advances in Neural Information Processing Systems*, 2017, pp. 1772–1783.

[47] P. Putzky and M. Welling, "Recurrent inference machines for solving inverse problems," *arXiv preprint arXiv:1706.04008*, 2017.

[48] M. Mardani, H. Monajemi, V. Papyan, S. Vasanawala, D. Donoho, and J. Pauly, "Recurrent generative adversarial networks for proximal learning and automated compressive image recovery," *arXiv preprint arXiv:1711.10046*, 2017.

[49] J. Adler and O. Öktem, "Learned primal-dual reconstruction," *IEEE transactions on medical imaging*, vol. 37, no. 6, pp. 1322–1332, 2018.

[50] Z. Deng, H. Zhang, X. Liang, L. Yang, S. Xu, J. Zhu, and E. P. Xing, "Structured generative adversarial networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 3899–3909.

[51] http://yann.lecun.com/exdb/mnist/, *Mnist database of handwritten digits*.

[52] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, 2006.

[53] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, Dec. 2015.

[54] K. R. Davidson and S. J. Szarek, "Local operator theory, random matrices and banach spaces," *Handbook of the geometry of Banach spaces*, vol. 1, no. 317-366, p. 131, 2001.

[55] R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices," *arXiv preprint arXiv:1011.3027*, 2010.

[56] A. S. Bandeira, R. Van Handel, *et al.*, "Sharp nonasymptotic bounds on the norm of random matrices with independent entries," *The Annals of Probability*, vol. 44, no. 4, pp. 2479–2506, 2016.

[57] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 677–695, 1997.

[58] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[59] W. Mantzel, J. Romberg, and K. Sabra, "Compressive matched-field processing," *The Journal of the Acoustical Society of America*, vol. 132, no. 1, pp. 90–102, 2012.

[60] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. E. Kelly, R. G. Baraniuk, *et al.*, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, p. 83, 2008.

[61] G. A. ten Holt, M. J. Reinders, and E. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in *Thirteenth annual conference of the Advanced School for Computing and Imaging*, vol. 300, 2007.

[62] A. Akl and S. Valaee, "Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing," in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 2010, pp. 2270–2273.

[63] Y. Shi and T. Tsui, "An fpga-based smart camera for gesture recognition in hci applications," in *Asian Conference on Computer Vision*, Springer, 2007, pp. 718–727.

[64] C.-T. Li and W.-H. Chen, "A novel fpga-based hand gesture recognition system.," *Journal of Convergence Information Technology*, vol. 7, no. 9, 2012.

[65] K. Wang, T. Gasser, *et al.*, "Alignment of curves by dynamic time warping," *The annals of Statistics*, vol. 25, no. 3, pp. 1251–1276, 1997.

[66] J. M. Carmona and J. Climent, "A performance evaluation of hmm and dtw for gesture recognition," in *Iberoamerican Congress on Pattern Recognition*, Springer, 2012, pp. 236–243.

[67] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.

[68] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[69] L. Liu and L. Shao, "Learning discriminative representations from rgb-d video data.," in *IJCAI*, vol. 1, 2013, p. 3.

[70] H. Wang, F Wang, H. Nguyen, S Li, T. Huang, A. Ahmed, M. Smith, N. Mannem, and J Lee, "Power amplifiers performance survey 2000-present," *[Online]. Available:* `https://gems.ece.gatech.edu/PA_survey.html`", 2018.

[71] F. Wang and H. Wang, "A 24-30GHz Watt-level broadband linear Doherty power amplifier with multi-primary distributed-active-transformer power-combining supporting 5G NR FR2 64QAM with ¿19 dBm average Pout and ¿19% average PAE," in *IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, 2020.

[72] ——, "An instantaneously broadband ultra-compact highly linear PA with compensated distributed Balun output network achieving ¿17.8dBm P1dB and ¿36.6% PAEP1dB over 24-40GHz and continuously supporting 64-/256-QAM 5G NR signals over 24-42GHz," in *IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Papers*, 2020.

[73] S. Hu, F. Wang, and H. Wang, "A 28-/37-/39-GHz linear Doherty power amplifier in silicon for 5G applications," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1586–1599, 2019.

[74] F. Wang, T.-W. Li, and H. Wang, "4.8 a highly linear super-resolution mixed-signal Doherty power amplifier for high-efficiency mm-wave 5G multi-Gb/s communications," in *2019 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2019, pp. 88–90.

[75] S. Hu, S. Kousai, and H. Wang, "Antenna impedance variation compensation by exploiting a digital doherty power amplifier architecture," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, no. 2, pp. 580–597, 2015.

[76] F. Wang, T.-W. Li, S. Hu, and H. Wang, "A super-resolution mixed-signal Doherty power amplifier for simultaneous linearity and efficiency enhancement," *IEEE Journal of Solid-State Circuits*, 2019.

[77] T. Kanar, S. Zihir, and G. M. Rebeiz, "A 2–15-GHz accurate built-in-self-test system for wideband phased arrays using self-correcting eight-state I/Q mixers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 12, pp. 4250–4261, 2016.

[78] F. Wang, S. Xu, J. Romberg, and H. Wang, "An artificial-intelligence (AI) assisted mm-wave Doherty power amplifier with rapid mixed-mode in-field performance optimization," in *Proc. IEEE IMC-5G*, 2019.

[79] A. Goyal, M. Swaminathan, A. Chatterjee, D. C. Howard, and J. D. Cressler, "A new self-healing methodology for RF amplifier circuits based on oscillation principles," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 10, pp. 1835–1848, 2011.

[80] J. Y.-C. Liu, A. Tang, N.-Y. Wang, Q. J. Gu, R. Berenguer, H.-H. Hsieh, P.-Y. Wu, C. Jou, and M.-C. F. Chang, "A V-band self-healing power amplifier with adaptive feedback bias control in 65 nm CMOS," in *2011 IEEE Radio Frequency Integrated Circuits Symposium*, IEEE, 2011, pp. 1–4.

[81] J. Y.-C. Liu, R. Berenguer, and M.-C. F. Chang, "Millimeter-wave self-healing power amplifier with adaptive amplitude and phase linearization in 65-nm CMOS," *IEEE Transactions on microwave theory and techniques*, vol. 60, no. 5, pp. 1342–1352, 2012.

[82] C. Chien, A. Tang, F. Hsiao, and M.-C. F. Chang, "Dual-control self-healing architecture for high-performance radio SoCs," *IEEE Design & Test of Computers*, vol. 29, no. 6, pp. 40–51, 2012.

[83] S. M. Bowers, K. Sengupta, K. Dasgupta, B. D. Parker, and A. Hajimiri, "Integrated self-healing for mm-wave power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 61, no. 3, pp. 1301–1315, 2013.

[84] H. Jia, B. Chi, L. Kuang, and Z. Wang, "Simple and robust self-healing technique for millimetre-wave amplifiers," *IET Circuits, Devices & Systems*, vol. 10, no. 1, pp. 37–43, 2016.

[85] N. Wang, V. Yousefzadeh, D. Maksimovic, S. Pajic, and Z. B. Popovic, "60% efficient 10-GHz power amplifier with dynamic drain bias control," *IEEE transactions on Microwave Theory and Techniques*, vol. 52, no. 3, pp. 1077–1081, 2004.

[86] Y. Noh and C. S. Park, "An intelligent power amplifier MMIC using a new adaptive bias control circuit for w-cdma applications applications," *IEEE Journal of solid-state circuits*, vol. 39, no. 6, pp. 967–970, 2004.

[87] W. Tai, H. Xu, A. Ravi, H. Lakdawala, O. Bochobza-Degani, L. R. Carley, and Y. Palaskas, "A transformer-combined 31.5 dBm outphasing power amplifier in 45 nm LP CMOS with dynamic power control for back-off power efficiency enhancement," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 7, pp. 1646–1658, 2012.

[88] F. Wang, S. Xu, J. Romberg, and H. Wang, "An artificial-intelligence (AI) assisted mm-wave multi-band Doherty transmitter," in *GOMACTech*, 2019.

[89] O. Hammi, S. Bousnina, and F. M. Ghannouchi, "A linearized doherty amplifier using complex baseband digital predistortion driven by CDMA signals," in *Proceedings. 2004 IEEE Radio and Wireless Conference (IEEE Cat. No. 04TH8746)*, IEEE, 2004, pp. 435–438.

[90] J. C. Cahuana, P. Landin, D. Gustafsson, C. Fager, and T. Eriksson, "Linearization of dual-input Doherty power amplifiers," in *2014 International Workshop on Integrated Nonlinear Microwave and Millimetre-wave Circuits (INMMiC)*, IEEE, 2014, pp. 1–3.

[91] M. Sajedin, I. Elfergani, J. Rodriguez, R. Abd-Alhameed, and M. F. Barciela, "A survey on RF and microwave Doherty power amplifier for mobile handset applications," *Electronics*, vol. 8, no. 6, p. 717, 2019.

[92] M. Xiao, "Novel predistortion techniques for rf power amplifiers," PhD thesis, University of Birmingham, 2009.

[93] N. B. Carvalho and J. C. Pedro, "Multi-tone intermodulation distortion performance of 3rd order microwave circuits," in *IEEE MTT-S International Microwave Symposium Digest*, vol. 2, 1999, pp. 763–766.

[94] A. A. Saleh, "Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers," *IEEE Transactions on communications*, vol. 29, no. 11, pp. 1715–1720, 1981.

[95] J. Vuolevi, T. Rahkonen, and J. Manninen, "Measurement technique for characterizing memory effects in rf power amplifiers," in *Radio and Wireless Conference, 2000. RAWCON 2000. 2000 IEEE*, IEEE, 2000, pp. 195–198.

[96] H. Ku, M. D. McKinley, and J. S. Kenney, "Quantifying memory effects in rf power amplifiers," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 12, pp. 2843–2849, 2002.

[97] H. Ku and J. S. Kenney, "Behavioral modeling of nonlinear RF power amplifiers considering memory effects," *IEEE transactions on microwave theory and techniques*, vol. 51, no. 12, pp. 2495–2504, 2003.

[98] C. Eun and E. J. Powers, "A new volterra predistorter based on the indirect learning architecture," *IEEE transactions on signal processing*, vol. 45, no. 1, pp. 223–227, 1997.

[99] L. Ding, R. Raich, and G. T. Zhou, "A Hammerstein predistortion linearization design based on the indirect learning architecture," in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, IEEE, vol. 3, 2002, pp. III–2689.

[100] S. Afsardoost, T. Eriksson, and C. Fager, "Digital predistortion using a vector-switched model," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 4, pp. 1166–1174, 2012.

[101] P. Goetz, S. Kumar, and R. Miikkulainen, "On-line adaptation of a signal predistorter through dual reinforcement learning," in *ICML*, 1996, pp. 175–181.

[102] A. S. Ciminski, "Recurrent neural networks usefulness in digital pre-distortion of power amplifiers," in *Microwaves, Radar and Wireless Communications, 2004. MIKON-2004. 15th International Conference on*, IEEE, vol. 1, 2004, pp. 249–252.

[103] B. Mulliez, E. Moutaye, H. Tap, L. Gatet, and F. Gizard, "Predistortion system implementation based on analog neural networks for linearizing high power amplifiers transfer characteristics," in *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, IEEE, 2013, pp. 412–416.

[104] P. M. Tomé, "Analog neural predistortion of power amplifiers," Master's thesis, Universidade de Aveiro, 2016.

[105] M. Sonal, *Machine learning for papr distortion reduction in ofdm systems*, 2016.

[106] H. Qian, H. Huang, and S. Yao, "A general adaptive digital predistortion architecture for stand-alone rf power amplifiers," *IEEE Transactions on Broadcasting*, vol. 59, no. 3, pp. 528–538, 2013.

[107] N. Dawar, T. Sharma, R. Darraji, and F. M. Ghannouchi, "Linearisation of radio frequency power amplifiers exhibiting memory effects using direct learning-based adaptive digital predistoriton," *IET Communications*, vol. 10, no. 8, pp. 950–954, 2016.

[108] J. Chani-Cahuana, P. N. Landin, C. Fager, and T. Eriksson, "Iterative learning control for RF power amplifier linearization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 9, pp. 2778–2789, 2016.

[109] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[110] X. Yang, D. Liu, and D. Wang, "Reinforcement learning for adaptive optimal control of unknown continuous-time nonlinear systems with input constraints," *International Journal of Control*, vol. 87, no. 3, pp. 553–566, 2014.

[111] Q. Zhao, H. Xu, and S. Jagannathan, "Near optimal output feedback control of nonlinear discrete-time systems based on reinforcement neural network learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 4, pp. 372–384, 2014.

[112] D. Abel, E. C. Williams, S. Brawner, E. Reif, and M. L. Littman, "Bandit-based solar panel control," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[113] H. D. Moura, D. F. Macedo, and M. A. Vieira, "Automatic quality of experience management for WLAN networks using multi-armed bandit," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 279–288.

[114] N. Wanigasekara, J. Schmalfuss, D. Carlson, and D. S. Rosenblum, "A bandit approach for intelligent IoT service composition across heterogeneous smart spaces," in *Proceedings of the 6th International Conference on the Internet of Things*, ACM, 2016, pp. 121–129.

[115] A. Ferdowsi, S. Ali, W. Saad, and N. B. Mandayam, "Cyber-physical security and safety of autonomous connected vehicles: Optimal control meets multi-armed bandit learning," *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7228–7244, 2019.

[116] K. M. Jagodnik, P. S. Thomas, A. J. van den Bogert, M. S. Branicky, and R. F. Kirsch, "Training an actor-critic reinforcement learning controller for arm movement using human-generated rewards," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 10, pp. 1892–1905, 2017.

[117] X. Yang, P. Yu, L. Feng, F. Zhou, W. Li, and X. Qiu, "A deep reinforcement learning based mechanism for cell outage compensation in 5g UDN," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 476–481.

[118] J. Lu, L. Li, J. Nguyen, D. Shen, X. Tian, G. Chen, and K. Pham, "Machine learning based adaptive predistorter for high power amplifier linearization," in *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, IEEE, 2019, pp. 1–6.

[119] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[120] R. Agrawal, "The continuum-armed bandit problem," *SIAM journal on control and optimization*, vol. 33, no. 6, pp. 1926–1951, 1995.

[121] R. D. Kleinberg, "Nearly tight bounds for the continuum-armed bandit problem," in *Advances in Neural Information Processing Systems*, 2005, pp. 697–704.

[122] E. W. Cope, "Regret and convergence bounds for a class of continuum-armed bandit problems," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1243–1253, 2009.

[123] A. J. Mersereau, P. Rusmevichientong, and J. N. Tsitsiklis, "A structured multiarmed bandit problem and the greedy policy," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2787–2802, 2009.

[124] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011.

[125] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, Springer, 2010, pp. 177–186.

[126] P. H. Calamai and J. J. Moré, "Projected gradient methods for linearly constrained problems," *Mathematical programming*, vol. 39, no. 1, pp. 93–116, 1987.

[127] A. Iusem, "On the convergence properties of the projected gradient method for convex optimization," *Computational & Applied Mathematics*, vol. 22, no. 1, pp. 37–52, 2003.

[128] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[129] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[130] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[131] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[132] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7-9, pp. 1180–1190, 2008.

[133] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," *arXiv preprint arXiv:1205.4839*, 2012.

[134] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 201–212, 2011.

[135] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[136] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.